RAPPORT INTERNE INT 122/85

TROISIEME CYCLE DE LA PHYSIQUE
EN SUISSE ROMANDE

Cours de base
Semestre d'hiver 1982/83

METODES NUMERIQUES EN PHYSIQUE

Chapitres III et VI.

R. GRUBER

CHAPITRE III CALCUL MATRICIEL

3.1 INTRODUCTION

La plupart des problèmes physiques peuvent être décrits par un système d'équations aux dérivées partielles. En général, ces systèmes ne peuvent être résolus analytiquement que dans des cas particuliers. Le cas général, lui, demande une résolution numérique. Le domaine est souvent discrétisé et l'opérateur différentiel est approximé localement par des différences finies ou par des éléments finis si l'on considère la représentation faible. De telles discrétisations conduisent à des problèmes matriciels: issus soit de problèmes d'équations linéaires ou non linéaires, soit de problèmes à valeurs propres.

Dans ce chapitre nous présentons des méthodes aussi "efficaces" que possible pour la résolution des problèmes matriciels. Par "efficace" nous entendons l'emploi de la meilleure méthode numérique en investissant le minimum de son propre temps: meilleur rapport qualité/temps. Cela peut conduire à un conflit, puisque souvent la meilleure méthode ne se programme pas facilement.

Nous allons tout d'abord présenter l'exemple particulier de l'équation de l'équilibre magnétohydrodynamique (MHD) idéal. Cette équation a une solution analytique (mais non physique) que nous utiliserons pour comparer les différentes méthodes. Celles-ci se divisent en méthodes directes et méthodes itératives. La méthode directe la plus courante est l'élimination de Gauss. Nous verrons que cette méthode garde la structure de bande d'une matrice, mais elle détruit les zéros à l'intérieur de la bande. Dans de gros problèmes bi- ou tridimensionnels, les bandes des matrices sont très creuses. Le remplissage de la bande d'une matrice symétrique et définie positive peut être évité en choisissant une méthode itérative.

Le système d'équations linéaires $A\underline{x} = \underline{b}$ où A est définie positive peut être remplacé par le problème: trouver le minimum de la forme bilinéaire $W = 1/2(A\underline{x},\underline{x}) - (\underline{b},\underline{x})$. Toutes les méthodes itératives sont basées sur cette idée. Dans la méthode de Gauss-Seidel on varie une composante du vecteur après l'autre de telle façon que W devient minimal pour chacune d'elles. Cette méthode converge toujours, mais très lentement. Au lieu d'aller en direction de la pente maximale on peut

aussi choisir la direction conjuguée. On parle alors de la méthode du gradient conjugué. Cette méthode est efficace si les surfaces à W = constant ne sont pas trop elliptiques. Afin d'éviter ce genre de problèmes, on essaie de préconditionner la matrice, ce qui revient à concentrer toutes les valeurs propres de la matrice autour de 1. Comme exemple de préconditionnement, citons la méthode de la décomposition incomplète.

Celui qui croit que la méthode de la direction du gradient (Gauss-Seidel) ne se pratique plus a tort. Au contraire, cette méthode présente l'avantage que l'on résoud localement l'équation et que l'on n'a jamais vraiment besoin d'une matrice. On peut en plus l'améliorer par une surrelaxation: on ne s'arrête pas au minimum, mais on remonte la pente. Lors de l'itération, on remarque que les erreurs de courtes longueurs d'ondes se répartissent rapidement, tandis que les erreurs globales à grandes longueurs d'ondes, ne disparaissent que lentement. La raison vient du fait suivant: le dernier point du réseau ne ressent la modification issue du premier qu'après un balayage complet du réseau. L'idée récente de la méthode des réseaux multiples (Multigrid) attaque précisément ce problème: une erreur globale ne doit pas être dispersée en itérant sur un réseau fin. Cela peut se faire sur un réseau grossier. Une fois l'erreur globale éliminée, on peut revenir sur le réseau fin. Cette méthode s'est montrée très efficace dans des situations idéales.

Pour la résolution de systèmes linéaires, la méthode directe de l'élimination de Gauss se trouve dans les librairies mathématiques comme NAG ou Linpack. Les méthodes iterative doivent être programmées.

Chercher les valeurs propres d'une matrice est un vieux problème et bien supporté par les librairies. Chercher les valeurs propres du problème $A\underline{x} = \lambda B\underline{x}$, par contre, n'est pas encore suffisamment documenté. Les méthodes proposées sont l'iteration inverse du vecteur et l'algorithme de Lanczos.

Dans la discussion du software, nous essayerons d'anticiper l'avenir: les programmes devraient déjà être prévus pour une utilisation sur une machine vectorielle, d'autant plus que l'expérience a montré un gain de 10 à 30% en temps lorsque des programmes écrits pour une machine vectorielle sont exécutés sur une machine scalaire. En plus, ils deviennent plus lisibles.

3.2 CREATION D'UNE MATRICE

3.2.1 UN EXEMPLE D'APPLICATION: L'EQUILIBRE MHD

Considérons un récipient torique dans lequel se trouve un gaz très chaud (à ~ 100 million de degrés). Ce gaz est entièrement ionisé: on le nomme "plasma".Afin d'éviter le contact plasma-parois, on utilise des champs magnétiques hélicoïdaux qui maintiennent le plasma dans un état d'équilibre. Cet équilibre est décrit par les équations MHD qui résultent de la description fluide du plasma. Les équations MHD sont:

$$\begin{aligned}
&\frac{D\rho}{Dt} = -\rho\,\underline{\nabla}\cdot\underline{v} && \text{(Continuité)}\\
&\rho\frac{D\underline{v}}{Dt} = -\underline{\nabla}p + \underline{J}\times\underline{B} && \text{(Newton)}\\
&\frac{D}{Dt}(\rho p^{-\gamma}) = 0 && \text{(état)}\\
&\underline{\nabla}\times\underline{B} = 0 && \\
& && \text{(Maxwell)}\\
&\underline{\nabla}\cdot\underline{B} = 0 && \\
&\frac{\partial\underline{B}}{\partial t} = \underline{\nabla}\times(\underline{v}\times\underline{B}) && \text{(loi d'Ohm)}
\end{aligned} \qquad (3.1)$$

Dans ce modèle idéal, on a admis que la resistivité est faible due à la haute température du gaz. Les grandeurs ρ, $\underline{v}$, p, γ, $\underline{J}$ et $\underline{B}$ sont la densité, la vitesse, la pression, l'adiabaticité, le courant et le champ magnétique. Nous utilisons le système d'unités naturelles de Weibel. Dans ce système, toutes les grandeurs sont de l'ordre de 1, en particulier $\mu_o = c = 1$. La vitesse $\underline{v}$, par exemple, est normalisée à la vitesse d'Alfvén $\underline{v}^2_A = \underline{B}^2/\rho$, qui est la vitesse caractéristique des ondes magnétohydrodynamiques. L'opérateur

$$\frac{D}{Dt} \equiv \frac{\partial}{\partial t} + \underline{v}\cdot\underline{\nabla} \qquad (3.2)$$

est la dérivée convective.

Dans un équilibre statique, on admet que toutes les grandeurs sont indépendantes du temps et que la vitesse est nulle. Les seules équations qui restent sont

$$\begin{aligned} \nabla p &= J \times B \\ J &= \nabla \times B \\ \nabla \cdot B &= 0 \end{aligned} \qquad (3.3)$$

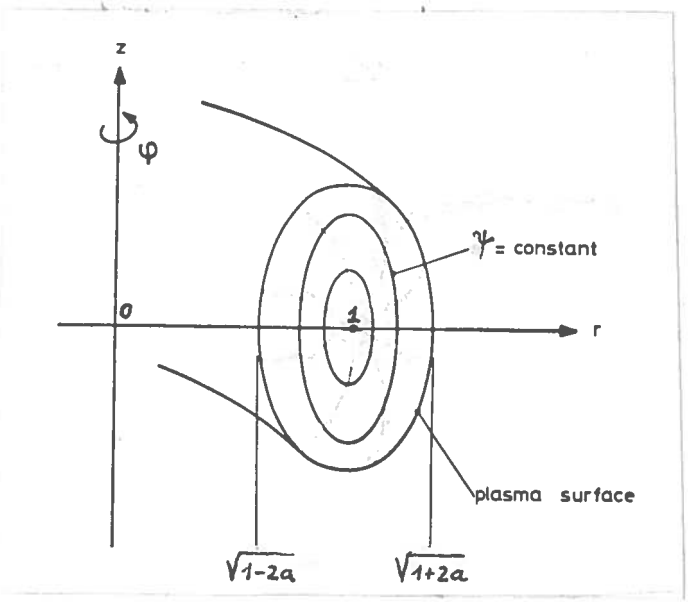
En géométrie axisymétrique torique (par exemple un Tokamak) B peut être écrit comme

$$B = T(\psi)\, \nabla \phi + \nabla\psi \times \nabla \phi \qquad (3.4)$$

où ϕ est l'angle torique (Fig. 3.1) le long duquel l'équilibre est inchangé.

Figure 3.1:

Système de coordonnées



Avec ce choix de $\underline{B}$ qui satisfait $\underline{\nabla} \cdot \underline{B} = 0$, il est simple de démontrer que $\underline{J} \times \underline{B}$ n'a qu'une composante le long $\underline{\nabla}\psi$, donc normale aux surfaces ψ = constante. Comme conséquence, $p = p(\psi)$ et l'équation de l'équilibre axisymétrique s'écrit

$$\mathcal{L}(\psi) = r \frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial \psi}{\partial r}\right) + \frac{\partial^2 \psi}{\partial z^2} = s(\psi, \rho) \equiv - r^2 \frac{dp}{d\psi} - T\frac{dT}{d\psi}. \qquad (3.5)$$

Dans le cas particulier $dp/d\psi = p'_0$ = constant et T = constant, dont $TdT/d\psi = 0$, la solution peut être trouvée analytiquement

$$\psi = \frac{\psi_s}{a^2}\left[\frac{r^2 z^2}{E^2} + (r^2 - 1)^2/4\right]$$

$$p'_0 = - \frac{2\psi_s(1+E^2)}{a^2 E^2} \qquad (3.6)$$

La surface du plasma est donnée par $\psi = \psi_s$:

$$\frac{r^2 z^2}{E^2} + (r^2 - 1)^2/4 = a^2, \qquad (3.7)$$

où E et a mesurent l'élongation et le petit rayon. La distance entre l'axe magnétique ($\psi = 0$, $z = 0$) et l'axe de symétrie est $r = 1$.

3.2.2 DISCRETISATION PAR DIFFERENCES FINIES

La méthode des différences finies est une méthode très souvent utilisée par le physicien pour approximer l'opérateur en un point. Choisissons d'abord une discrétisation du domaine (Fig. 3.2), équidistante en r et en z. En un point du réseau ($r = r_i$, $z = z_j$), l'inconnue est $\psi = \psi_{ij}$. Les différentes parties de l'opérateur $\mathcal{L}$ au point (r_i, z_j) sont approchées par

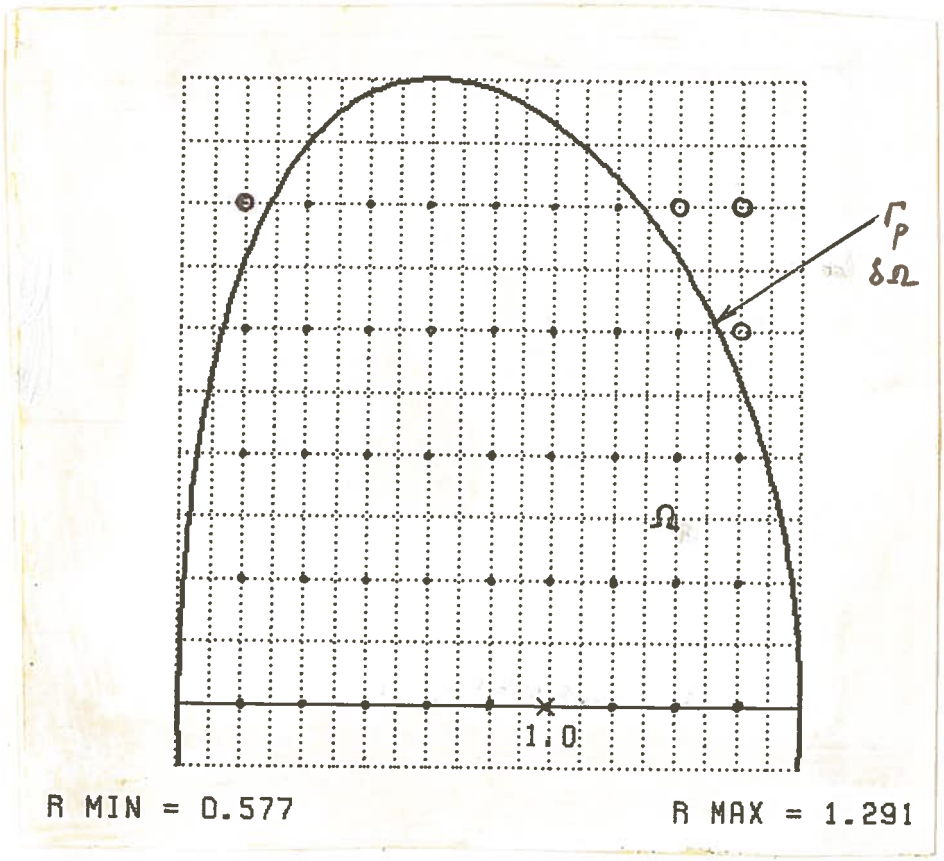


Figure 3.2:

Maillage en r et en z. N_r=20, N_z=11. Pour la matrice les points du bord du réseau ne sont pas considérés. Les points plus prononcés (. à l'intérieur, o à l'extérieur du domaine) correspondent à un maillage avec N_r=10, N_z=6 (voir figure 3.3).

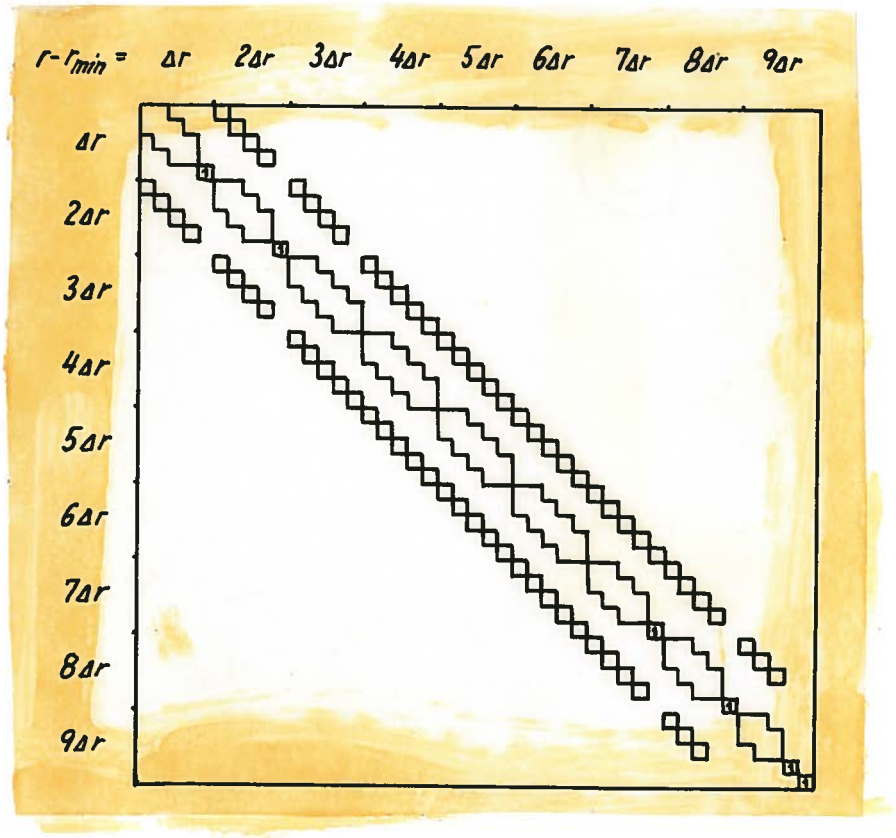


Figure 3.3:

Structure de la matrice A pour N_r=10, N_z=6 (points prononcés dans figure 3.2). La numérotation des points se fait d'abord le long z, donc en rangées de r = constant. Les 1 sur la diagonale correspondent à des points à l'extérieur du domaine.

$$\frac{\partial^2\psi}{\partial r^2} = \frac{\psi_{i-1j} - 2\,\psi_{ij} + \psi_{i+1j}}{\Delta r^2} + O(\Delta r^2)$$

$$\frac{\partial^2\psi}{\partial z^2} = \frac{\psi_{ij-1} - 2\,\psi_{ij} + \psi_{ij+1}}{\Delta z^2} + O(\Delta z^2) \qquad (3.8)$$

$$\frac{\partial\psi}{\partial r} = \frac{\psi_{i+1j} - \psi_{i-1j}}{2\,\Delta r} + O(\Delta r^2)$$

L'équation (3.5) s'écrit au point (r_i, z_j):

$$\mathcal{L}(r_i,z_j) = \frac{1}{\Delta z^2}\psi_{ij-1} + \left(\frac{1}{\Delta r^2} + \frac{1}{2r\Delta r}\right)\psi_{i-1j} - \left(\frac{2}{\Delta r^2} + \frac{2}{\Delta z^2}\right)\psi_{ij} + \left(\frac{1}{\Delta r^2} - \frac{1}{2r\Delta r}\right)\psi_{i+1j}$$

$$+ \frac{1}{\Delta z^2}\psi_{ij+1} = S(r_i) \qquad (3.9)$$

Cette discrétisation doit être modifiée pour les points proches de la surface $\psi = \psi_S$. Pour ces points, les valeurs exactes de la surface sont utilisées. On approxime localement la fonction ψ par un polynôme d'ordre 2 à partir duquel on calcule les dérivées. Notons que l'approximation de l'opérateur pour ces points n'est plus que $O(\Delta r) + O(\Delta z)$.

3.2.3 LE PROBLEME MATRICIEL

Pour construire la matrice A qui représente l'opérateur $\mathcal{L}$ discretisé, nous subdivisons le domaine en $N_r = 20$ et $N_z = 20$ intervalles équidistants en r et z. La surface du plasma ($\psi = \psi_S$) donnée par l'équation (3.7) est nommée Γ_p. En imposant la symétrie haut-bas, on admet donc que $\psi(z) = \psi(-z)$. Cela réduit le nombre d'intervalles en z de 20 à 11 (voir Figure 3.2). On choisit 11 au lieu de 10 afin de pouvoir imposer la symétrie $\psi(z_1) = \psi(z_3)$. Pour le cas $N_r = 10$,

N_Z = 6, la matrice A qui est non symétrique à cause du terme $(\partial\psi/\partial r)/r$ est donnée dans la figure 3.3 si l'on numérote les inconnues en rangées de r = constant. Cela diminue la largeur de bande qui est de 21 au lieu de 39 si l'on commence à numéroter le long de r. On voit que seules 5 rangées parallèles ont des valeurs non nulles. Il est donc possible de compacter cette matrice comme une bande de largeur 5 en se rappelant à quelle rangée dans la matrice initiale une colonne appartient. Le long de la diagonale apparaissent un certain nombre de 1 qui correspondent à des zéros dans le membre de droite S. Ceci survient chaque fois qu'un point du réseau est situé à l'extérieur du plasma. Nous avons donc attribué à tous les points du réseau une inconnue ψ_{ij}. Celles qui se trouvent à l'extérieur du réseau sont mises à zéro en biffant la ligne et la colonne, mettant 1 sur la diagonale et forçant la composante du membre de droite à zéro. La résolution matricielle donne alors ψ_{ij} = 0 pour tous les points à l'extérieur du plasma. Nous avons fait cela afin de garder la structure de la matrice inchangée. Si on éliminait ces lignes et colonnes pour le cas N_r = 20 et N_z = 11, on obtiendrait une matrice de longueur 163, au lieu de 190, avec une largeur de bande variable.

3.3 PROPRIETES D'UNE MATRICE

3.3.1 STRUCTURES DE MATRICES

A partir de la discrétisation de l'équation de l'équilibre MHD (3.5), nous avons vu que l'on obtient une matrice très creuse. Cette matrice A peut être décrite de plusieurs manières:

(a) matrice pleine de taille $A(N_r * N_z, N_r * N_z)$

(b) matrice de bande de taille $A(2*N_z+1, N_r*N_z)$ ou $A(2*N_r+1, N_r*N_z)$ dépendant de la numérotation

(c) matrice de bande creuse de taille $A(N_r*N_z,5)$.

Si l'on double N_r et N_z, la taille de la matrice augmente par un facteur 16 pour une matrice pleine, par un facteur 8 pour une matrice de bande et par un facteur de 4 pour une matrice creuse. Notons que la représentation d'une matrice de bande creuse devient compliquée si nous éliminons les points à l'extérieur du plasma.

3.3.2 DECOMPOSITION D'UNE MATRICE

Une matrice quadratique régulière peut être décomposée en un produit de 3 matrices

$$A = LDR \tag{3.10}$$

où L est une matrice triangulaire gauche (left) ou inférieure, D est une matrice diagonale et R est une matrice triangulaire droite (right) ou supérieure. Les matrices L et R ont des 1 sur la diagonale. Donc le déterminant de A sera égal au déterminant de D. Si A est symétrique, $R = L^T$ et la décomposition s'écrit

$$A = LDL^T. \tag{3.11}$$

La matrice A symétrique est définie positive si tous les éléments de D sont positifs. Cette propriété résulte du théorème de Sylvestre qui dit que le nombre de valeurs propres négatives d'une matrice A ne change pas lors d'une transformation $A = \hat{M}AM^T$ où M est une matrice quelconque régulière. La décomposition (3.11) correspond à une telle transformation avec M = L, et $\hat{A} = D$.

Des décompositions comme celles données par les équations (3.10 et 3.11) sont utilisées pour résoudre des systèmes d'équations linéaires. Elles permettent en outre de tester si une matrice est régulière ou si une matrice symétrique est définie positive.

Il faut encore noter que L a la même structure que la partie gauche de A et R la même que la partie droite de A.

Nous présentons la décomposition (3.10) qui se fait entièrement à l'intérieur de A par intermédiaire d'un exemple

$$A = \overbrace{\begin{pmatrix} a_{11} & a_{12} & a_{13} & & & \\ a_{21} & a_{22} & a_{23} & a_{24} & & \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & \\ & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\ & & a_{53} & a_{54} & a_{55} & a_{56} \\ & & & a_{64} & a_{65} & a_{66} \end{pmatrix}}^{MR}_{ML}$$

$$L = \underbrace{\begin{pmatrix} 1 & & & & & \\ \ell_{21} & 1 & & & & \\ \ell_{31} & \ell_{32} & 1 & & & \\ & \ell_{42} & \ell_{43} & 1 & & \\ & & \ell_{53} & \ell_{54} & 1 & \\ & & & \ell_{64} & \ell_{65} & 1 \end{pmatrix}}_{ML}$$

$$D = \begin{pmatrix} d_1 & & & & & \\ & d_2 & & & & \\ & & d_3 & & & \\ & & & d_4 & & \\ & & & & d_5 & \\ & & & & & d_6 \end{pmatrix}$$

$$R = \overbrace{\begin{pmatrix} 1 & r_{12} & r_{13} & & & \\ & 1 & r_{23} & r_{24} & & \\ & & 1 & r_{34} & r_{35} & \\ & & & 1 & r_{45} & r_{46} \\ & & & & 1 & r_{56} \\ & & & & & 1 \end{pmatrix}}^{MR}$$

$$DR = \begin{bmatrix} d_1 & d_1 r_{12} & d_1 r_{13} & & & \\ & d_2 & d_2 r_{23} & d_2 r_{24} & & \\ & & d_3 & d_3 r_{34} & d_3 r_{35} & \\ & & & d_4 & d_4 r_{45} & d_4 r_{46} \\ & & & & d_5 & d_5 r_{56} \\ & & & & & d_6 \end{bmatrix}$$

$$LDR = \left[\begin{array}{c|c|c|c|c|c} d_1 & d_1 r_{12} & d_1 r_{13} & & & \\ \ell_{21} d_1 & \ell_{21} d_1 r_{12} + d_2 & \ell_{21} d_1 r_{13} + d_2 r_{23} & d_2 r_{24} & & \\ \ell_{31} d_1 & \ell_{31} d_1 r_{12} + \ell_{32} d_2 & \ell_{31} d_1 r_{13} + \ell_{32} d_2 r_{23} + d_3 & \ell_{32} d_2 r_{24} + d_3 r_{34} & d_3 r_{35} & \\ & \ell_{42} d_2 & \ell_{42} d_2 r_{23} + \ell_{43} d_3 & \ell_{42} d_2 r_{24} + \ell_{43} d_3 r_{34} + d_4 & \ell_{43} d_3 r_{35} + d_4 r_{45} & d_4 r_{46} \\ & & \ell_{53} d_3 & \ell_{53} d_3 r_{34} + \ell_{54} d_4 & \ell_{53} d_3 r_{35} + \ell_{54} d_4 r_{45} + d_5 & \ell_{54} d_4 r_{46} + d_5 r_{56} \\ & & & \ell_{64} d_4 & \ell_{64} d_4 r_{45} + \ell_{65} d_5 & \ell_{64} d_4 r_{46} + \ell_{65} d_5 r_{56} + d_6 \end{array}\right]$$

Identifions A = LDR :

1) $d_1 = a_{11}$ (= pivot)
$\ell_{21} = a_{21}/d_1$
$\ell_{31} = a_{31}/d_1$
$r_{12} = a_{12}/d_1$
$r_{13} = a_{13}/d_1$

2) $a_{22} = a_{22} - \ell_{21} d_1 r_{12}$
$a_{32} = a_{32} - \ell_{31} d_1 r_{12}$
$a_{23} = a_{23} - \ell_{21} d_1 r_{13}$
$a_{33} = a_{33} - \ell_{31} d_1 r_{13}$

ML*MR opérations

3) $d_2 = a_{22}$ (= pivot)
$\ell_{32} = a_{32}/d_2$
$\ell_{42} = a_{42}/d_2$
$r_{23} = a_{23}/d_2$
$r_{24} = a_{24}/d_2$

4) $a_{33} = a_{33} - \ell_{32} d_2 r_{23}$
$a_{43} = a_{43} - \ell_{42} d_2 r_{23}$
$a_{34} = a_{34} - \ell_{32} d_2 r_{24}$
$a_{44} = a_{44} - \ell_{42} d_2 r_{24}$

ML*MR opérations

5) $d_3 = a_{33}$ (= pivot)
$\ell_{43} = a_{43}/d_3$
$\ell_{53} = a_{53}/d_3$
$r_{34} = a_{34}/d_3$
$r_{35} = a_{44}/d_3$

6) $a_{44} = a_{44} - \ell_{43} d_3 r_{34}$
$a_{54} = a_{54} - \ell_{53} d_3 r_{34}$
$a_{45} = a_{45} - \ell_{43} d_3 r_{35}$
$a_{55} = a_{55} - \ell_{53} d_3 r_{35}$

ML*MR opérations

7) $d_4 = a_{44}$ (= pivot)
$\ell_{54} = a_{54}/d_4$
$\ell_{64} = a_{64}/d_4$
$r_{45} = a_{45}/d_4$
$r_{46} = a_{46}/d_4$

8) $a_{55} = a_{55} - \ell_{54} d_4 r_{45}$
$a_{65} = a_{65} - \ell_{64} d_4 r_{45}$
$a_{56} = a_{56} - \ell_{54} d_4 r_{46}$
$a_{66} = a_{66} - \ell_{64} d_4 r_{46}$

ML*MR opérations

9) $d_5 = a_{55}$ (= pivot)
$\ell_{65} = a_{65}/d_5$
$r_{56} = a_{56}/d_5$

10) $a_{66} = a_{66} - \ell_{65} d_5 r_{56}$

(ML-1)*(MR-1) opérations

11) $d_6 = a_{66}$ (= pivot)

1er exemple

$$A = \begin{array}{|cccccc|}
5 & -5 & 10 & & & \\
15 & -13 & 36 & 14 & & \\
10 & -12 & 15 & -16 & 4 & \\
 & 8 & 19 & 70 & -40 & 12 \\
 & & 2 & 0 & -13 & 6 \\
 & & & 32 & -157 & 117
\end{array}$$

1) + 2)

$$\begin{array}{|cccccc|}
5 & -1 & 2 & & & \\
3 & 2 & 6 & 14 & & \\
2 & -2 & -5 & -16 & 4 & \\
 & 8 & 19 & 70 & -40 & 12 \\
 & & 2 & 0 & -13 & 6 \\
 & & & 32 & -157 & 117
\end{array}$$

3) + 4)

$$\begin{array}{|cccccc|}
5 & -1 & 2 & & & \\
3 & 2 & 3 & 7 & & \\
2 & -1 & 1 & -2 & 4 & \\
 & 4 & -5 & 14 & -40 & 12 \\
 & & 2 & 0 & -13 & 6 \\
 & & & 32 & -157 & 117
\end{array}$$

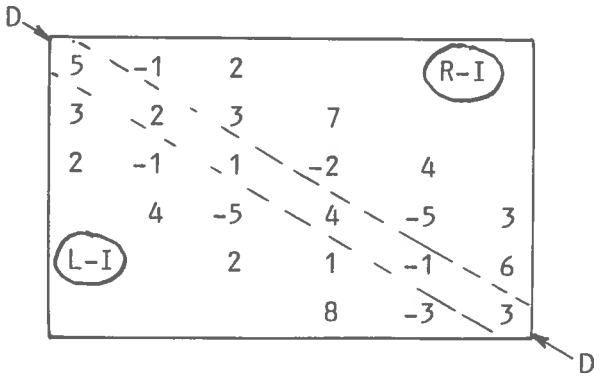
5) + 6)

$$\begin{array}{|cccccc|}
5 & -2 & 2 & & & \\
3 & 2 & 3 & 7 & & \\
2 & -1 & 1 & -2 & 4 & \\
 & 4 & -5 & 4 & -20 & 12 \\
 & & 2 & 4 & -21 & 6 \\
 & & & 32 & -157 & 117
\end{array}$$

7) + 8)

5	-1	2			
3	2	3	7		
2	-1	1	-2	4	
	4	-5	4	-5	3
		2	1	-1	-6
			8	3	21

9) + 10)
+ 11)



(I = matrice d'identité)

2ème exemple

$$A = \begin{pmatrix} 2 & 4 & 0 \\ 6 & 12 & 5 \\ 0 & 10 & 5 \end{pmatrix}$$

1) + 2)

2	2	5
3	0	5
0	10	5

pivot $a_{22} = 0$!

Mais: A n'est pas singulière !

3.3.3 CHOIX DU PIVOT

Il n'est pas nécessaire de prendre un pivot après l'autre sur la diagonale. Pour minimiser l'erreur d'arrondi, on peut choisir l'élément le plus grand en valeur absolue par ligne ou par colonne.

a) Pivoting par ligne

$$A = \begin{bmatrix} 2 & 4 & 0 \\ 6 & 12 & 5 \\ 0 & 10 & 5 \end{bmatrix}$$

1) + 2)

$$\begin{bmatrix} 1/2 & (4) & 0 \\ 0 & 3 & 5 \\ -5 & 5/2 & 1 \end{bmatrix}$$

3) + 4) + 5)

$$\begin{bmatrix} 1/2 & (4) & 0 \\ 0 & 3 & (5) \\ (-5) & 5/2 & 1 \end{bmatrix}$$

pivots choisis: 4 , 5, -5

No de colonne → 2 3 1

Réarrangement :
pour que les pivots
tombent sur la diagonale

$$\begin{bmatrix} (4) & 0 & 1/2 \\ 3 & (5) & 0 \\ 5/2 & 1 & (-5) \end{bmatrix}$$

Contrôle

No de colonne → 2 3 1

$$\begin{bmatrix} 1 & & \\ 3 & 1 & \\ 5/2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 4 & & \\ & 5 & \\ & & -5 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1/2 \\ & 1 & 0 \\ & & 1 \end{bmatrix} = \begin{bmatrix} 4 & 0 & 2 \\ 12 & 5 & 6 \\ 10 & 5 & 0 \end{bmatrix}$$

Revenir sur
la numérotation
initiale :

$$\begin{bmatrix} 2 & 4 & 0 \\ 6 & 12 & 5 \\ 0 & 10 & 5 \end{bmatrix} = A$$

b) Pivoting par colonne

$$A = \begin{bmatrix} 2 & 4 & 0 \\ 6 & 12 & 5 \\ 0 & 10 & 5 \end{bmatrix}$$

1) + 2)

$$\begin{bmatrix} 1/3 & 0 & -10/6 \\ (6) & 2 & 5/6 \\ 0 & 10 & 5 \end{bmatrix}$$

3) + 4)
+ 5)

$$\begin{bmatrix} 1/3 & 0 & (-10/6) \\ (6) & 2 & 5/6 \\ 0 & (10) & 1/2 \end{bmatrix}$$

pivots choisis : 6 , 10 , -10/6

Réarrangement :
pour que les valeurs de D
tombent sur la diagonale

$$\begin{bmatrix} (6) & 2 & 5/6 \\ 0 & (10) & 1/2 \\ 1/3 & 0 & (-10/6) \end{bmatrix} \quad \begin{matrix} 2 \\ 3 \\ 1 \end{matrix}$$

No de ligne

Contrôle

$$\begin{bmatrix} 1 & & \\ 0 & 1 & \\ 1/3 & 0 & 1 \end{bmatrix} \begin{bmatrix} 6 & & \\ & 10 & \\ & & -10/6 \end{bmatrix} \begin{bmatrix} 1 & 2 & 5/6 \\ & 1 & 1/2 \\ & & 1 \end{bmatrix} = \begin{bmatrix} 6 & 12 & 5 \\ 0 & 10 & 5 \\ 2 & 4 & 0 \end{bmatrix} \quad \begin{matrix} 2 \\ 3 \\ 1 \end{matrix}$$

No de ligne

Numérotation initiale :

$$\begin{bmatrix} 2 & 4 & 0 \\ 6 & 12 & 5 \\ 0 & 10 & 5 \end{bmatrix} = A$$

3.3.4 SOFTWARE ET APPLICATIONS

Pour résoudre un système d'équations linéaires, $A\underline{x} = \underline{b}$, où A est une matrice de bande non symétrique, nous utilisons trois librairies de programmes: MATLIB, LINPACK et NAG.

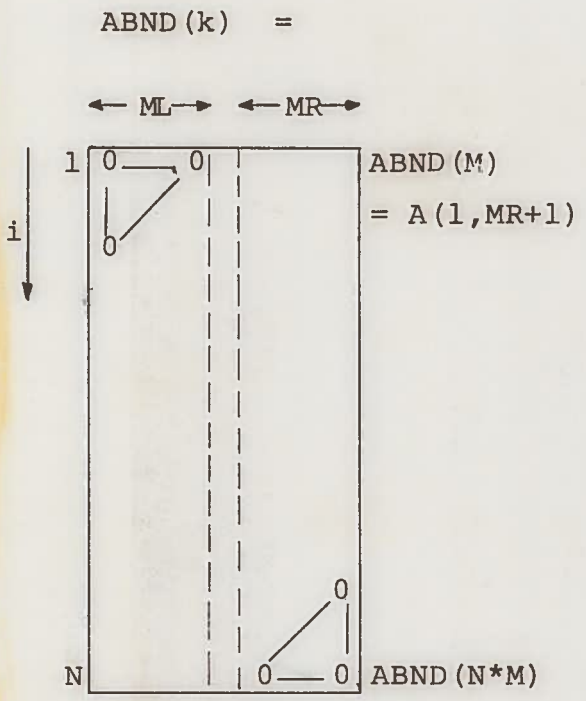
(a) MATLIB (CRPP-EPFL)

- Nom du sous-programme: BALDR (Banded A = LDR)
- Dimension de A : A = A (M*N), M = ML + MR + 1
- Manière de stocker A est montrée à la Figure 3.4.
- Pas de pivoting
- Test pour pivot(i) = 0 : idiag = M*(i-1) + ML + 1
  if (Abs(A(idiag) - A(idiag-1)*A(idiag-M)*A(idiag-M+1)) .LT.
  EPSMAC*Abs(A(idiag))) Nsing = -1
  (EPSMAC mesure la précision de la machine, donc EPSMAC ~ 10^{-14} pour CDC et EPSMAC ~ 10^{-7} pour VAX)
- Vectorisation garantie par l'utilisation des sous-programmes CRAY1, comme SAXPY, SCOPY, SDOT, SSCAL, SXYPZ, VZERO
- Temps CPU sur CDC 170-720 pour M = 21, N = 190: .293"
- Place mémoire pour A : M*N mots

(b) LINPACK (SIAM)

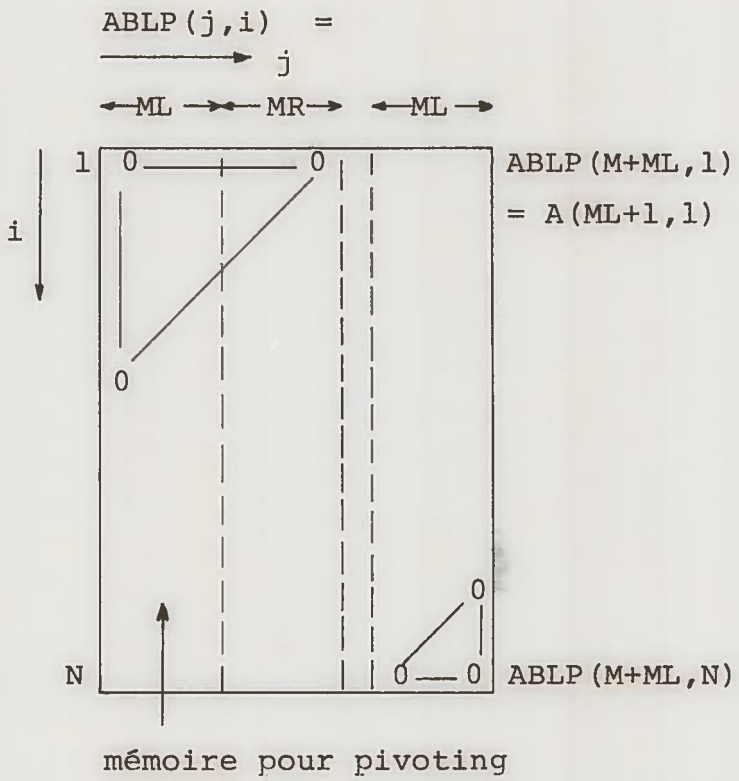
- Nom du sous-programme: SGBCO ou SGBFA
- Dimension de A: A = A(M + ML, N)
- Manière de stocker A est montrée à la Figure 3.5.
- Pivoting par colonne. C'est la raison pour laquelle ML rangées diagonales doivent être ajoutées à droite de A.
- Test pour pivot(i) = 0:
  SGBCO : RCOND = 0
  SGBFA : INFO = i

MATRICE A DE BANDE (M = ML+MR+1)



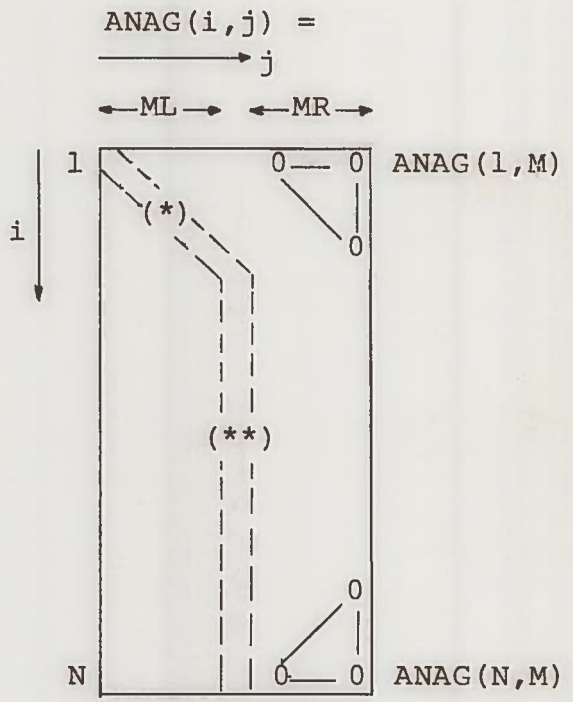
Diagonale:
ABND(M*(i-1)+ML+1)

FIG. 3.4 MATLIB



Diagonale:
ABLP(M,i)

FIG. 3.5 LINPACK



Diagonale:
(*) ANAG(i,i) $1 \leqslant i \leqslant ML+1$
(**) ANAG(i,ML+1) $ML+1 \leqslant i \leqslant N$

FIG. 3.6 NAG

- Vectorisation garantie par l'utilisation des sous-programmes CRAY1
- Temps CPU pour CDC 170-720 pour M = 21, N = 190 : .360"
- Place mémoire pour A : (M + ML)*N mots
- Pour le problème de l'équilibre MHD idéal, le pivot choisi est toujours la diagonale.
- La valeur de RCOND ≈ $|\lambda max|/|\lambda min|$ qui mesure la condition de A peut être très grande dû au mauvais conditionnement de A ou dû à des échelles différentes dans le problème.

(c) NAG

- Nom du sous-programme: F01BMF
- Dimension de A: A = A(M,N)
- Manière de stocker A est montrée à la Figure 3.6
- Tableau additionnel : AL(ML,N)
- Pivoting par colonne. C'est la raison pour laquelle le tableau AL est nécessaire
- Test pour pivot(i) = 0: IFAIL = i
- Source ne peut être obtenue
- Temps CPU pour CDC 170-720 pour M = 21, N = 190 : .539"
- Place mémoire pour A et AL: (M+ML)*N mots
- Pour le problème de l'équilibre MHD idéal, le pivot choisi est toujours la diagonale.

MATLIB, LINPACK et NAG effectuent donc exactement le même calcul.

3.4 $\underline{A}\underline{x} = \underline{b}$: METHODES DIRECTES

3.4.1 INVERSION

Une méthode pour résoudre le système d'équations linéaires $\underline{A}\underline{x} = \underline{b}$ est de multiplier $\underline{b}$ avec l'inverse de A :

$$\underline{x} = A^{-1}\,\underline{b} \qquad (3.12)$$

Cela n'est possible que si A est une matrice régulière. Malheureusement l'inverse d'une matrice creuse est en général une matrice pleine. Les seules exceptions sont la matrice diagonale dont l'inverse reste diagonale et la matrice triangulaire gauche ou droite et creuse dont l'inverse est aussi une matrice triangulaire gauche ou droite mais pleine. Donc, n'inversez jamais une matrice creuse!

3.4.2 ELIMINATION DE GAUSS

Toutes les librairies modernes résolvent un système d'équations linéaires $\underline{A}\underline{x} = \underline{b}$ en deux pas :

(a) Décomposition de A

$$A = LDR \qquad (3.13)$$

(b) Résolution de

$$LDR\underline{x} = \underline{b} \qquad (3.14)$$

La partie (a) a déjà été discutée dans le chapitre 3.2. Pour le problème $LDT\underline{x} = \underline{b}$, nous appelons

$$\begin{aligned} \underline{w} &= R\underline{x} \\ \underline{v} &= DR\underline{x} \end{aligned} \qquad (3.15)$$

et la résolution s'écrit:

$$\begin{aligned} L\underline{v} &= \underline{b} \\ D\underline{w} &= \underline{v} \\ R\underline{x} &= \underline{w} \end{aligned} \qquad (3.16)$$

Souvent, les deux premiers systèmes peuvent être combinés en un seul $LD\underline{w} = \underline{b}$ ce qui consiste à résoudre un système de bande triangulaire gauche de haut en bas. Le troisième système peut être exécuté une fois les deux premiers achevés. Il consiste à effectuer une résolution d'un système de bande triangulaire droite de bas en haut.

Pour de différentes structures de la matrice A, différents programmes dans MATLIB, LINPACK et NAG doivent être appelés. Pour notre problème d'application, nous donnons avec les exercices les fiches descriptives pour LINPACK tandis que pour MATLIB, un listing des routines nécessaires est distribué.

Le temps de calcul pour une décomposition d'une matrice de longueur N et de largeur de bande M est proportionnel à $N * M^2$; minimisez donc la largeur de bande M! La résolution $LDR\ \underline{x} = \underline{b}$ ne prend qu'un nombre d'opérations proportionnel à N * M. La place mémoire prise par A est N * M mots pour MATLIB et environ 3/2 N * M mots pour LINPACK et NAG.

3.4.4 A SYMETRIQUE ET DEFINIE POSITIVE

En physique, beaucoup de problèmes peuvent être formulés en tant que problèmes variationnels. Le système d'équations linéaires que l'on obtient après discrétisation et différentiation par rapport à toutes les inconnues est symétrique et défini positif. Dans un tel problème, le système d'équations linéaires $A\underline{x} = \underline{b}$ peut être directement résolu par une décomposition de Cholesky

$$A = R^T R \qquad (3.17)$$

et une résolution du système

$$R^T\underline{w} = \underline{b}$$
$$R\underline{x} = \underline{w}. \qquad (3.18)$$

Au lieu de stocker toute la bande de largeur M = ML + MR + 1, il suffit de stocker MR + 1 colonnes. La décomposition se fait de la même façon que décrite dans le cas non-symétrique. Pour la résolution d'un système d'équations linéaires $A\underline{x} = \underline{b}$ où A est symétrique et définie positive, il existe des routines spéciales dans les trois librairies.

3.5 METHODES ITERATIVES

3.5.1 METHODE DE GAUSS-SEIDEL

Au lieu de résoudre le système

$$A\underline{x} = \underline{b}, \qquad (3.19)$$

où A est symétrique et définie positive par la méthode directe décrite dans le chapitre 3.4, nous pourrions itérer sur une solution approchée $\underline{v}$ de telle façon que le vecteur résiduel

$$\underline{r} = A\underline{v} - \underline{b} \qquad (3.20)$$

tende vers zéro. Puisque A est admise définie positive, ce problème peut aussi être formulé de manière équivalente par:

"Chercher le minimum de la forme quadratique

$$W(\underline{v})=1/2(A\underline{v},\underline{v})-(\underline{b},\underline{v})" \qquad (3.21)$$

On aimerait faire diminuer $W(\underline{v})$ en variant $\underline{v}$. Pour cette raison, nous choisissons une direction de relaxation $\underline{d}$ et nous varions le paramètre t du vecteur

$$\underline{v}' = \underline{v} + t\underline{d} \qquad (3.22)$$

de telle façon que

$$W(\underline{v}') = W(\underline{v}+t\underline{d}) = 1/2t^2(A\underline{d},\underline{d}) + t(\underline{r},\underline{d}) + W(\underline{v}) \qquad (3.23)$$

devienne minimal, donc

$$dW(\underline{v}')/dt = 0 = t(A\underline{d},\underline{d}) + (\underline{r},\underline{d}). \qquad (3.24)$$

Nous obtenons ainsi la valeur optimale de t pour la direction $\underline{d}$:

$$t = t_{opt} = -(\underline{r},\underline{d})/(A\underline{d},\underline{d}) \qquad (3.25)$$

Puisqu'il s'agit d'un problème quadratique en t et A est définie positive, W diminue pour $0 < t < 2t_{opt}$.

Quelles directions faut-il choisir pour le vecteur $\underline{d}$? Dans la méthode de Gauss-Seidel, on prend les vecteurs unités $\underline{e}_k$ où k = 1,..., N. Ceci revient à varier cycliquement une composante de $\underline{v}$ après l'autre. Pour le pas i, la composante v_i sera modifiée par

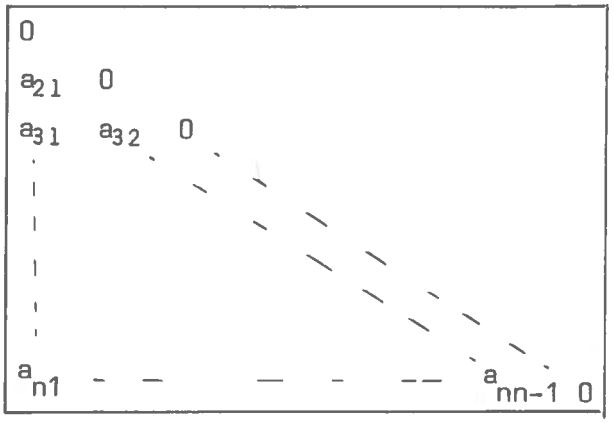
$$v'_i = v_i - \frac{r_i}{a_{ii}} = v_i - \left(\sum_{j=1}^{N} a_{ij} v_j - b_i\right)/a_{ii} \qquad (3.26)$$

Ce processus correspond à la décomposition de A en

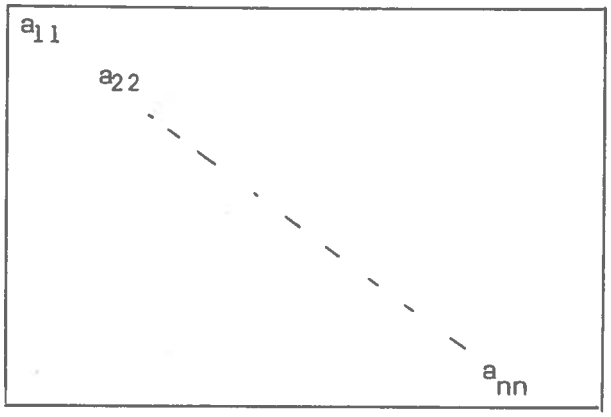
$$A = E + D + F \qquad (3.27)$$

où

$E =$

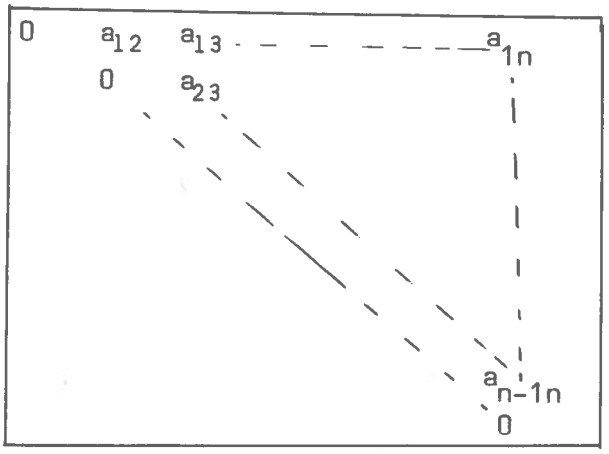


$D =$



et

$F =$



L'itération de Gauss-Seidel s'écrit alors:

$$D\underline{v}^{(k+1)} = -E\underline{v}^{(k+1)} - F\underline{v}^{(k)} + \underline{b} \qquad (3.28)$$

Cette itération converge toujours mais très lentement. Cela est spécialement vrai si les lignes de niveau W = constant, sont des ellipsoïdes très allongés. Voici un exemple d'un système d'équations linéaires qui montre bien cet effet:

$$\begin{pmatrix} 1 & 10 \\ 10 & 101 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} -1 \\ -11 \end{pmatrix} \qquad (3.29)$$

La solution est $x_1 = 9$ est $x_2 = -1$. La forme quadratique $W(\underline{v})$ correspondante s'écrit

$$W(\underline{v}) = 1/2\left[v_1+10v_2+1)^2 + (v_2+1)^2-2\right]. \qquad (3.30)$$

Le vecteur $\underline{v}$ est identique à $\underline{x}$ si

$$(\partial W/\partial v_1) = (\partial W/\partial v_2) = 0. \qquad (3.31)$$

Le minimum de W pour $v_1 = x_1 = 9$ et $v_2 = x_2 = -1$ est $W = -1$. Les surfaces W = constant sont des ellipses concentriques très allongés. Le centre est la solution (9,-1) et le rapport entre la grande et la petite axe est ~100. Il faut noter que pour $x_1 = 19$ et $x_2 = -2$ la distance à la solution vaut $\sqrt{101} \approx 10$ et $W = -1/2$. Pour $x_1 = 9$ et $x_2 = -2$ par contre, la distance de la solution vaut seulement 1 mais W = 49.5! Cela ne veut pas dire que l'estimation initiale (19,-2) est meilleure pour l'itération que celle en (9,-2).

Cette méthode peut aussi être appliquée à notre problème de l'équilibre MHD. Nous verrons au chapitre 6 que la forme variationnelle qui décrit l'équation (3.5) est définie positive. Malheureusement, la discrétisation par différences finies la rend non symétrique. Mais, plus on augmente le nombre d'intervalles, plus on symétrise A et à la limite pour $N_r = N_z = \infty$, A est symétrique et définie positive. L'avantage de la méthode de Gauss-Seidel est qu'on peut l'appliquer directement sur la description locale de l'opérateur, eq. (3.9). On s'y prend comme suit:

(1) Choisir une solution approchée initiale $\underline{v}^{(0)}$ qui peut être arbitraire ou le résultat d'un calcul d'un cas similaire.

(2) Résoudre cycliquement pour toutes les inconnues en avançant ou en direction de r, de z ou diagonalement.

La direction diagonale a l'avantage que les opérations ne sont pas récursives, donc vectorisables. Ce processus est itératif. Le nombre d'itérations dépend fortement de la condition de la matrice. En plus, le nombre d'itérations est proportionnel au nombre d'intervalles. Cela est dû aux erreurs globales de grande longueur d'onde. On peut montrer qu'une erreur globale n'est diminuée que par un facteur de $1-O(h^2)$ par itération. Par contre, les erreurs locales, donc de courte longueur d'onde, sont amorties par un grand facteur qui est ~4 pour une longueur d'onde de 2h.

3.5.2 GRADIENT CONJUGUE

Au lieu d'avancer le long de la direction normale aux surfaces W = constantes, on peut avancer le long de la direction conjuguée. Soit k le pas de l'itération, nous avons alors (voir Figure 3.7):

- direction normale $-\underline{r}^{(k)} = -A\underline{v}^{(k)}+\underline{b}$ (3.32)
- direction tangentielle $\underline{h}^{(k-1)}$
- direction conjuguée $\underline{h}^{(k)}$

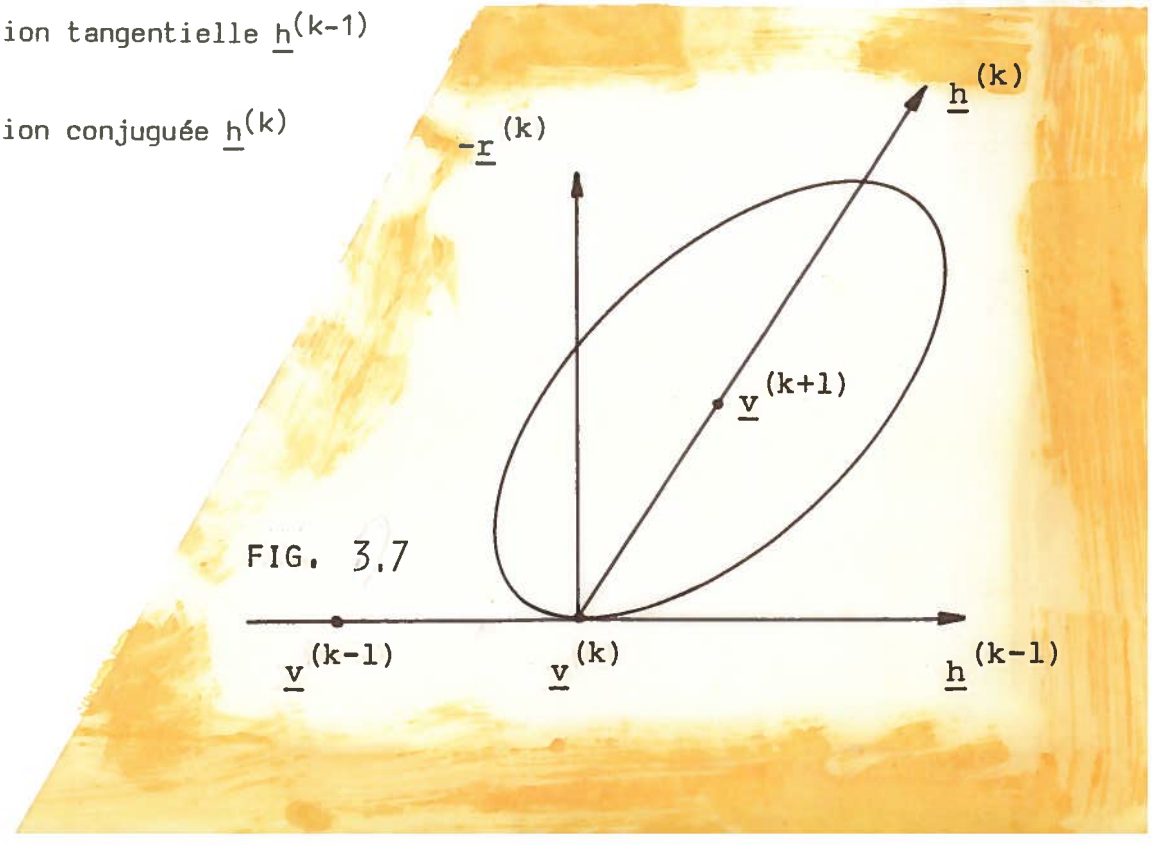


FIG. 3.7

La direction conjuguée

$$\underline{h}^{(k)} = -\underline{r}^{(k)} + \varepsilon^{(k)}\, \underline{h}^{(k-1)} \tag{3.33}$$

est reliée à la direction tangentielle $\underline{h}^{(k-1)}$ par

$$(A\underline{h}^{(k)}, \underline{h}^{(k-1)}) = (A\underline{h}^{(k-1)}, \underline{h}^{(k)}) = 0. \tag{3.34}$$

La nouvelle solution approchée

$$\underline{v}^{(k+1)} = \underline{v}^{(k)} + t^{(k)}\, \underline{h}^{(k)} \tag{3.35}$$

s'obtient en minimisant la forme biléaire par rapport à $t^{(k)}$

$$\frac{dW}{dt^{(k)}} = \frac{d}{dt^{(k)}} \{(A\underline{v}^{(k+1)}, \underline{v}^{(k+1)}) - (\underline{v}^{(k+1)}, \underline{b})\} = 0 \tag{3.36}$$

L'équation (3.34) définit le paramètre

$$\varepsilon^{(k)} = \frac{(\underline{r}^{(k)}, A\underline{h}^{(k-1)})}{(\underline{h}^{(k-1)}, A\underline{h}^{(k-1)})} \tag{3.37}$$

et l'équation (3.36) le paramètre $t^{(k)}$

$$t^{(k)} = \frac{-(\underline{h}^{(k)}, \underline{r}^{(k)})}{(\underline{h}^{(k)}, A\underline{h}^{(k)})} \tag{3.38}$$

L'effort numérique est principalement fourni pour le calcul de $A\underline{h}^{(k)}$. Le calcul de $r^{(k)}$ qui représenterait un effort numérique semblable à celui de $Ah^{(k)}$ peut être évité en imposant les conditions d'orthogonalité

$$(\underline{r}^{(k)}, \underline{h}^{(k-1)}) = (\underline{r}^{(k)}, \underline{r}^{(k-1)}) = 0 \tag{3.39}$$

Multipliant (3.33) avec $\underline{r}^{(k)}$ donne

$$(\underline{h}^{(k)}, \underline{r}^{(k)}) = -(\underline{r}^{(k)}, \underline{r}^{(k)}) + \varepsilon^{(k)}(\underline{h}^{(k-1)}, \underline{r}^{(k)}) = -(\underline{r}^{(k)}, \underline{r}^{(k)}) \qquad (3.40)$$

La définition du résidu $\underline{r}^{(k+1)}$ dans l'équation (3.32) peut être transformée en utilisant (3.35)

$$\underline{r}^{(k)} = A\underline{v}^{(k)} - \underline{b} = \underline{r}^{(k-1)} + t^{(k-1)} A\underline{h}^{(k-1)} \qquad (3.41)$$

et

$$(\underline{r}^{(k)}, A\underline{h}^{(k-1)}) = \frac{1}{t^{(k-1)}}(\underline{r}^{(k)}, \underline{r}^{(k)} - \underline{r}^{(k-1)}) = \frac{1}{t^{(k-1)}} (\underline{r}^{(k)}, \underline{r}^{(k)}) \qquad (3.42)$$

Le processus d'itération s'écrit alors :

(1) $k = 0$

choisir $\underline{v}^{(0)}$

$\underline{r}^{(0)} = A\underline{v}^{(0)} - \underline{b}$

$\underline{h}^{(0)} = -\underline{r}^{(0)}$

(2) $\underline{u}^{(k)} = A\underline{h}^{(k)}$

(3) $t^{(k)} = \|\underline{r}^{(k)}\|^2 / (\underline{h}^{(k)}, \underline{u}^{(k)})$

(4) $\underline{v}^{(k+1)} = \underline{v}^{(k)} + t^{(k)} \underline{h}^{(k)}$

(5) $\underline{r}^{(k+1)} = \underline{r}^{(k)} + t^{(k)} \underline{u}^{(k)}$

(6) $\varepsilon^{(k)} = \|\underline{r}^{(k+1)}\|^2 / \|\underline{r}^{(k)}\|^2$

(7) $\underline{h}^{(k+1)} = -\underline{r}^{(k+1)} + \varepsilon^{(k)} \underline{h}^{(k)}$

$k = k+1$

go to (2)

On peut démontrer que ce processus converge après au plus N itérations où N est le nombre d'équations. Néanmoins, à cause des erreurs d'arrondi, le nombre d'itérations est souvent plus grand. C'est justement une des raisons pour lesquelles la méthode du gradient conjugué n'a pas eu un grand succès. Une autre raison réside dans la place mémoire utilisée qui est de 10N (5N pour A, N pour $\underline{v}$, $\underline{b}$, $\underline{r}$, $\underline{h}$ et $\underline{u}$) comparé à 2N (N pour $\underline{v}$ et $\underline{b}$) pour la méthode du gradient.

3.5.3 GRADIENT CONJUGUE AVEC PRECONDITIONNEMENT

Afin de diminuer le nombre d'itérations de la méthode du gradient conjugué, on pourrait par exemple transformer la matrice A de façon que l'ellipticité des surfaces de niveau W = constant diminue. Par une telle transformation (préconditionnement), on cherche à rendre la matrice proche de l'identité: ce qui amènerait toutes les valeurs propres de A autour de 1.

Une des méthodes de préconditionnement les plus utilisées aujourd'hui est celle connue sous le nom de décomposition de "Choleski incomplète". Il s'agit de décomposer A approximativement en

$$A \approx R^T R \tag{3.43}$$

où R a la même structure que A. Pratiquement, on décompose A comme auparavant. Mais chaque fois que l'on rencontre un zéro dans A, on met un zéro dans R. Si un pivot devient négatif - dans une décomposition exacte, cela ne peut pas se produire puisqu'on admet que A est définie positive - on met un 1 sur la diagonale de R. La matrice

$$\tilde{A} = R^{-T} A R^{-1} \tag{3.44}$$

est une matrice proche de la matrice identité. Au lieu de résoudre $A\underline{x} = \underline{b}$, on résoud

$$\tilde{A}\underline{y} = \hat{\underline{b}}$$

où

$$\underline{y} = R\underline{x} \tag{3.45}$$

$$\hat{\underline{b}} = R^{-T}\underline{b}.$$

Pour notre problème de l'équilibre MHD, la matrice A n'est pas symétrique. Mais puisqu'elle représente un opérateur auto-adjoint, on peut quand-même appliquer la méthode du gradient conjugué. Le préconditionnement s'effectue en écrivant

$$\tilde{A} = L^{-1} A R^{-1} \tag{3.46}$$

où les diagonales de L et R sont identiques. On applique la même règle que pour Choleski incomplet. Au lieu de résoudre $A\underline{x} = \underline{b}$, on résoud

$$\tilde{A}\,\underline{y} = \hat{\underline{b}}$$

où

$$\underline{y} = R\underline{x} \tag{3.47}$$

$$\hat{\underline{b}} = L^{-1}\,\underline{b}.$$

Appliquons les trois méthodes itératives décrites à notre problème de l'équilibre MHD. La méthode de Gauss-Seidel converge (l'erreur relative de chaque composante est inférieure à 10^{-6}) après 348 pas d'itérations où une itération correspond à un passage à travers tout le réseau. En utilisant la méthode du gradient conjugé il faut toujours 163 itérations prenant le même temps. Avec préconditionnement, le nombre d'itérations descend à 19, en prenant seulement 1/5 du temps des autres méthodes. Puisque les valeurs propres sont ramenées autour de 1, un raffinement du maillage n'augmente guère le nombre d'itérations.

3.5.4 SURRELAXATION

La méthode du gradient peut être fortement accélerée par une surrelaxation. Cela veut dire que l'on ne s'arrête pas au minimum (voir équations 3.20 à 3.28) mais que l'on remonte la pente :$t > t_{opt}$. Donc au lieu de se contenter de $\underline{v}^{(k+1)}$ dans équation (3.28), on forme un autre vecteur

$$\tilde{\underline{v}}^{(k+1)} = \omega \underline{v}^{(k+1)} + (1-\omega)\,\underline{v}^{(k)} \tag{3.48}$$

où $1 \leq \omega < 2$. Pour $\omega = 1$, on retrouve la méthode du gradient.

La surrelaxation permet de venir à bout plus rapidement des erreurs globales donc de grande longueur d'onde. La valeur optimale $\omega = 1$, ($t = t_{opt}$), a été trouvée en distribuant l'erreur localement. Pour des erreurs globales on peut trouver un facteur de surrelaxation optimal, $\omega = \omega_{opt}$:

$$\omega_{opt} \tilde{=} 2 - c/\tilde{N} \tag{3.49}$$

où $N = \text{Max}(N_r, N_z)$. Plus on augmente N, plus il faut choisir ω_{opt} proche de 2. Pour notre problème de l'équilibre MHD

$$\omega_{opt} \tilde{=} 2 - 4.6/N_r. \tag{3.50}$$

Spécifiquement, pour $N_r = 20$, $\omega_{opt} = 1.77$. Dans la figure 3.8 le nombre d'itérations est donné en fonction de ω pour deux discrétisations $(N_r,N_z) = (20,10)$ et $(40,20)$. On voit que pour $\omega = 1$, le nombre d'itérations augmente par un facteur 4, l'erreur par itération diminue donc par un facteur $1 - O(h^2)$, tandis que pour le choix $\omega = \omega_{opt}$, le nombre d'itérations n'augmente que par un facteur 2, l'erreur par itération diminue donc par un facteur $1 - O(h)$. Il faut aussi noter que le nombre d'itérations augmente rapidement si l'on dépasse ω_{opt}. Il est préférable de choisir $\omega < \omega_{opt}$ plutôt que $\omega > \omega_{opt}$.

Pour le problème de l'équilibre MHD, le nombre d'itérations est de 43 pour $\omega = \omega_{opt} = 1.77$ et augmente proportionnel à N_r. La place mémoire utilisée dépend de l'opinion : recalculer pour chaque pas les éléments de la matrice A et du membre de droite ou non. Si l'on recalcule chaque fois toutes les grandeurs nécessaires pour résoudre l'équation localement, la place mémoire n'est que N mots (longueur du vecteur $\underline{v}$). Si l'on stocke une fois pour toute la matrice A et le membre de droite $\underline{b}$, la place mémoire est de 7N (5N pour A, N pour $\underline{b}$ et $\underline{v}$).

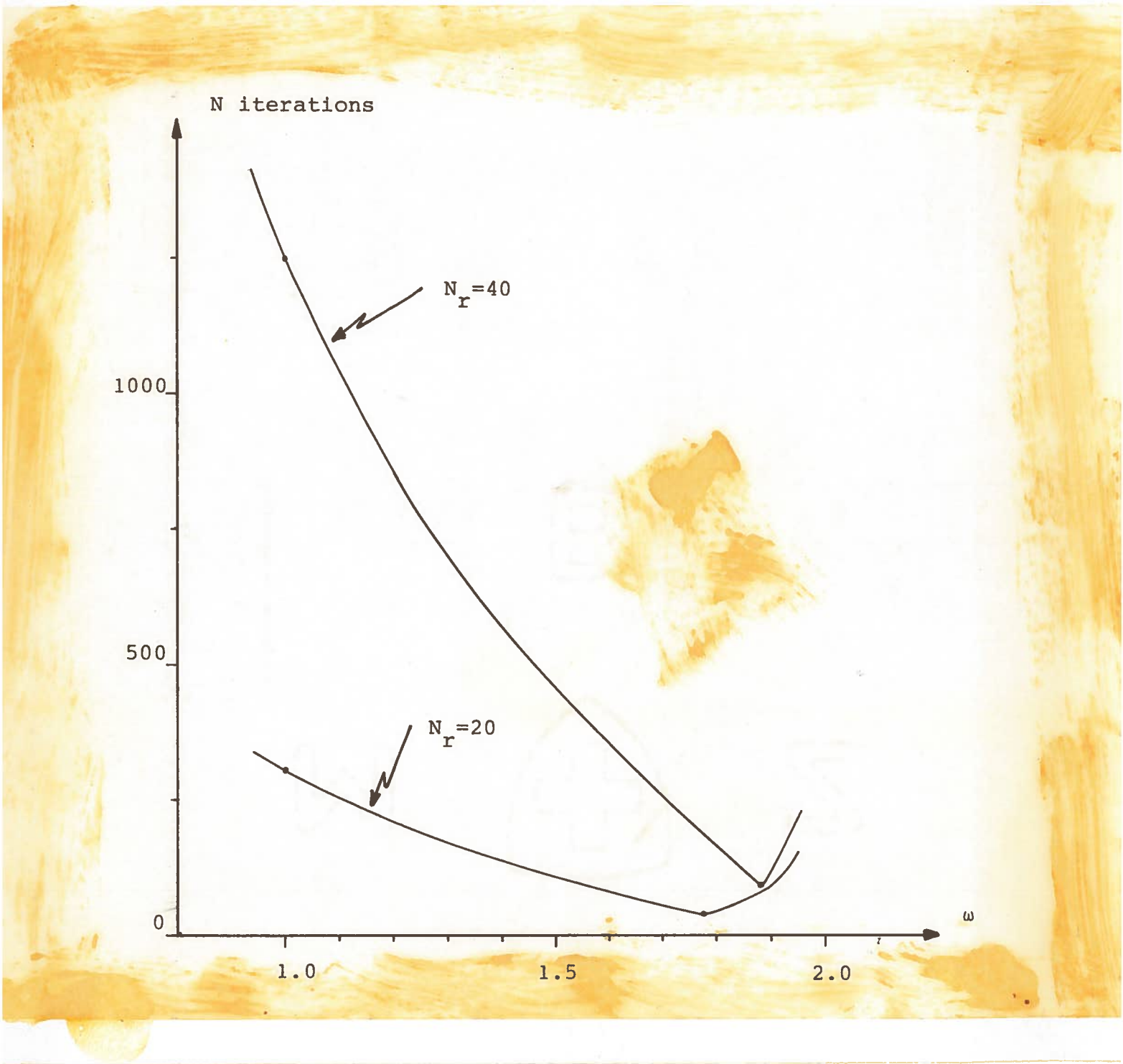


Fig. 3.8: Variation du nombre d'itérations N en fonction du paramètre de surrelaxation ω pour 2 réseaux différents.

3.5.5 MULTIGRID

La méthode de la surrelaxation converge en O(h) à cause des erreurs de grandes longueurs d'onde. Ainsi plus le maillage sera fin (N petit), plus la convergence sera lente. Alors pourquoi ne pas itérer à moindre frais avec un maillage plus grossier, où l'erreur de grandes longueurs d'onde s'éliminera plus rapidement et revenir ensuite au réseau fin ? Voilà rapidement esquissée l'idée maîtresse de la méthode à maillages multiples ou MULTIGRID.

Reprenons le problème différentiel

$$\mathcal{L}\,\phi(r,z) = S(\phi,r,z), \tag{3.51}$$

et approximons-le sur un maillage à $N = z^L$ intervalles par

$$\mathcal{L}^{(L)}\phi^{(L)}(r,z) = S^{(L)}(\phi^{(L)},r,z). \tag{3.52}$$

Lorsque l'opérateur $\mathcal{L}$ est discrétisé à l'aide des différences finies ou des éléments finis de plus bas ordre (éléments linéaires), la formulation approchée tend vers le problème exact par $O(h^2)$. On peut donc écrire:

$$\phi(r,z) - \phi^{(L)}(r,z) = O(2^{-2L}). \tag{3.53}$$

De plus, la solution $\phi^{(L)}$, qui approxime la solution exacte ϕ sur un maillage à 2^L intervalles (niveau L), est elle-même approchée de manière itérative par $u^{(L)}$. Le problème approximé se récrira sous la forme:

$$\mathcal{L}^{(L)}u^{(L)}(r,z) = S^{(L)}(u^{(L)},r,z) - R^{(L)}(u^{(L)},r,z) \tag{3.54}$$

où la grandeur $R^{(L)}$ représente le résidu, qui doit tendre vers zéro lorsque $u^{(L)}$ tend vers $\phi^{(L)}$. Notons $V^{(L)} = \phi^{(L)} - u^{(L)}$, la différence des solutions exactes et itératives. Soustrayons l'équation (3.54) de (3.52), nous obtenons une équation pour $V^{(L)}$:

$$\mathcal{L}^{(L)}V^{(L)} = R^{(L)}(V^{(L)},r,z) \tag{3.55}$$

L'idée de Multigrid est la suivante:

- On calcule la composante globale de $V^{(L)}$ sur un ou plusieurs maillages grossiers (L réduit) en résolvant

$$\mathcal{L}^{(L-k)} V^{(L-k)} = R^{(L-k)} . \qquad (3.56)$$

- Une fois l'erreur $V^{(L-k)}$ au niveau L-k connue, on interpole cette solution pour obtenir les composantes de $V^{(L)}$ sur le réseau fin, non sans oublier d'utiliser les propriétés de convergence de la solution $\phi^{(L)}$ pour l'extrapolation de l'erreur d'un niveau à l'autre. Si cette opération n'est pas réalisée avec soin, en allant du niveau L-k au niveau L on rajoute une erreur globale de l'ordre de $O(2^{-2(L-k)})-O(2^{-2L})$.

- De retour au niveau L, on corrige la solution approchée $u^{(L)}$ grâce à $V^{(L)}$. Les erreurs locales introduites par l'interpolation des composantes et l'extrapolation d'un niveau à l'autre, sont éliminées rapidement par une relaxation de Gauss-Seidel.

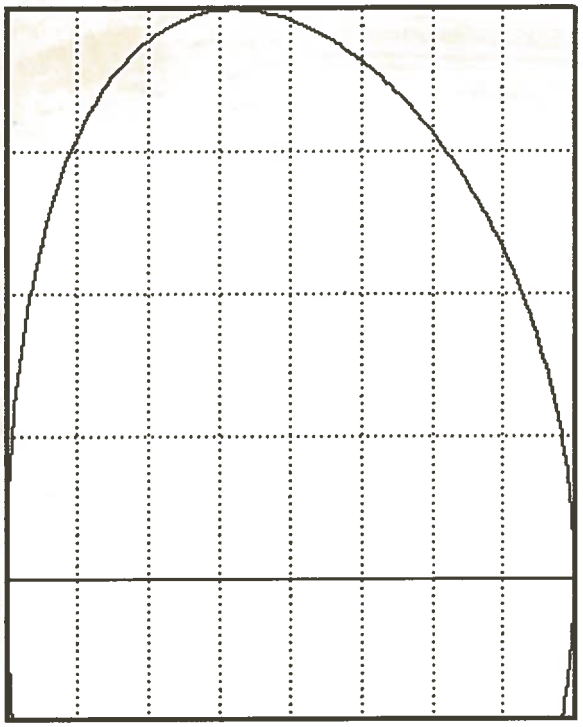
- Ce processus peut être répété jusqu'à ce que le résidu $||R^{(L)}|| < \varepsilon \, ||S^{(L)}||$.

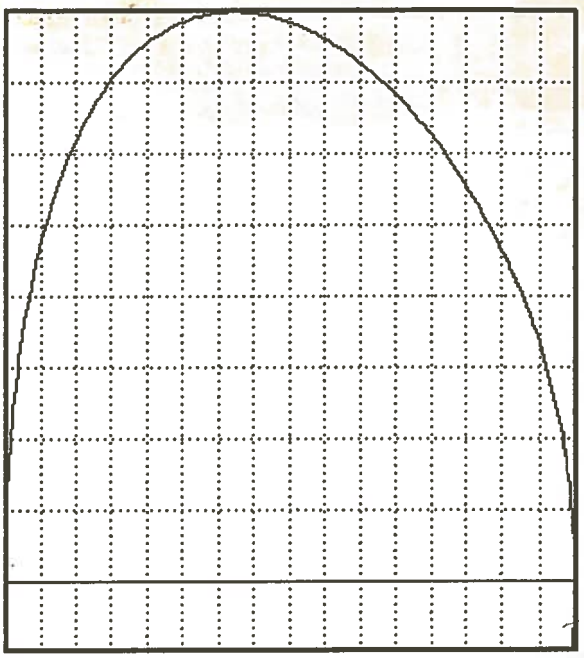
Nous avons appliqué une variante de Multigrid au calcul de l'équilibre MHD. Afin d'éliminer les difficultés du traitement de la surface du plasma Γ_p (voir figure 1), nous considérons la solution ϕ dans tout le rectangle en prescrivant les valeurs analytiques de ϕ sur le cadre. Nous avons choisi une séquence de 5 réseaux, correspondant à L = 3,4,5,6,7 pour NR et NZ = NR/2 (voir Figure 3.9).

A partir du maillage grossier L = 3, nous avançons vers le réseau le plus fin L = 7, en effectuant les pas suivants:

L = 3: A partir d'une solution initiale, 28 itérations par surrelaxation avec ω = 1.45 pour obtenir $\phi^{(3)}$

L = 4: Solution initiale à partir de $\phi^{(3)}$. Par interpolation cubique et 31 itérations par surrelaxation avec ω = 1.70 on obtient $\phi^{(4)}$





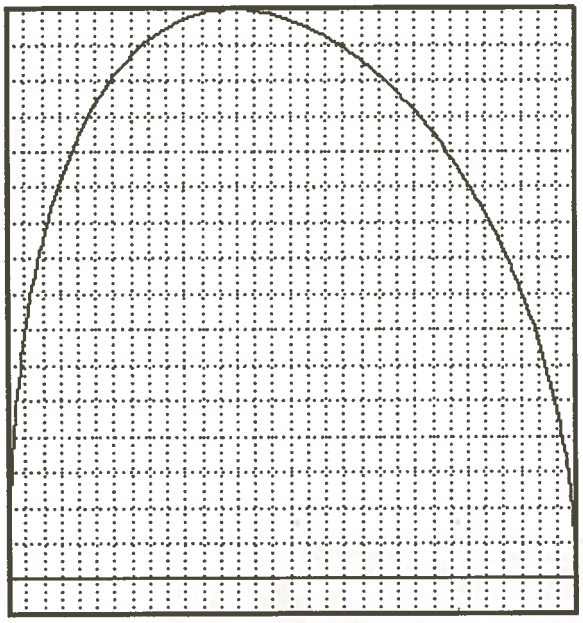


Fig. 3.9: Séquence MULTIGRID:

Représentation des maillages L = 3,4 et 5 de notre problème d'application.

L = 5: Solution initiale par extrapolation à partir de $\phi^{(3)}$ et $\phi^{(4)}$ sur les points du réseau communs aux trois niveaux L = 3, 4 et 5. On admet une convergence quadratique, donc $\phi^{(5)} = (5\phi^{(4)} - \phi^{(3)})/4$ pour 1/16 des points du niveau L = 5. Extrapolation des points du niveau L = 4 non communs avec L = 3 en admettant les mêmes dépendances fonctionnelles locales pour L = 4 et L = 5. Ceci remplit encore 3/16 des points du réseau du niveau L = 5. Interpolation cubique des points de L = 5 non communs avec L = 4. 37 itérations par surrelaxation avec ω = 1.80 pour distribuer les erreurs globales de l'ordre de $O(2^{-4L})$ provenant du niveau L = 3 et locales dues aux extrapolations et interpolations.

L = 6: Solution initiale analogue à L = 5 à partir des solutions $\phi^{(4)}$ et $\phi^{(5)}$. Il se revèle que l'erreur globale $O(2^{-4L})$ du niveau L = 4 est plus petite que l'erreur locale dûe aux interpolations et extrapolations. Afin de mieux distribuer les erreurs locales, une relaxation de Gauss-Seidel donne la meilleure convergence (ω = 1 et 11 itérations).

L = 7: Processus analogue à L = 6 à partir des solutions $\phi^{(5)}$ et $\phi^{(6)}$. Deux pas d'itération de Gauss-Seidel donnent déjà un résidu $||R^{(7)}|| < \varepsilon \, ||S^{(7)}||$ où $\varepsilon = 10^{-8}$.

Il n'est donc pas nécessaire de revenir sur un maillage grossier pour éliminer l'erreur globale du résidu.

Notons que la simple surrelaxation avec ω = 1.96 nécessite environ 200 itérations pour L = 7.

Fig. 3.10: Dépendance approximative du temps de calcul en fonction du nombre d'intervalles.

GS : Gauss-Seidel

GSS : GS + Surrélaxation

GC : Gradient Conjugué

IDCG: Décomposition Incomplète + Gradient Conjugué

MG : Multigrid

D : Direct Method (Gauss Elimination)

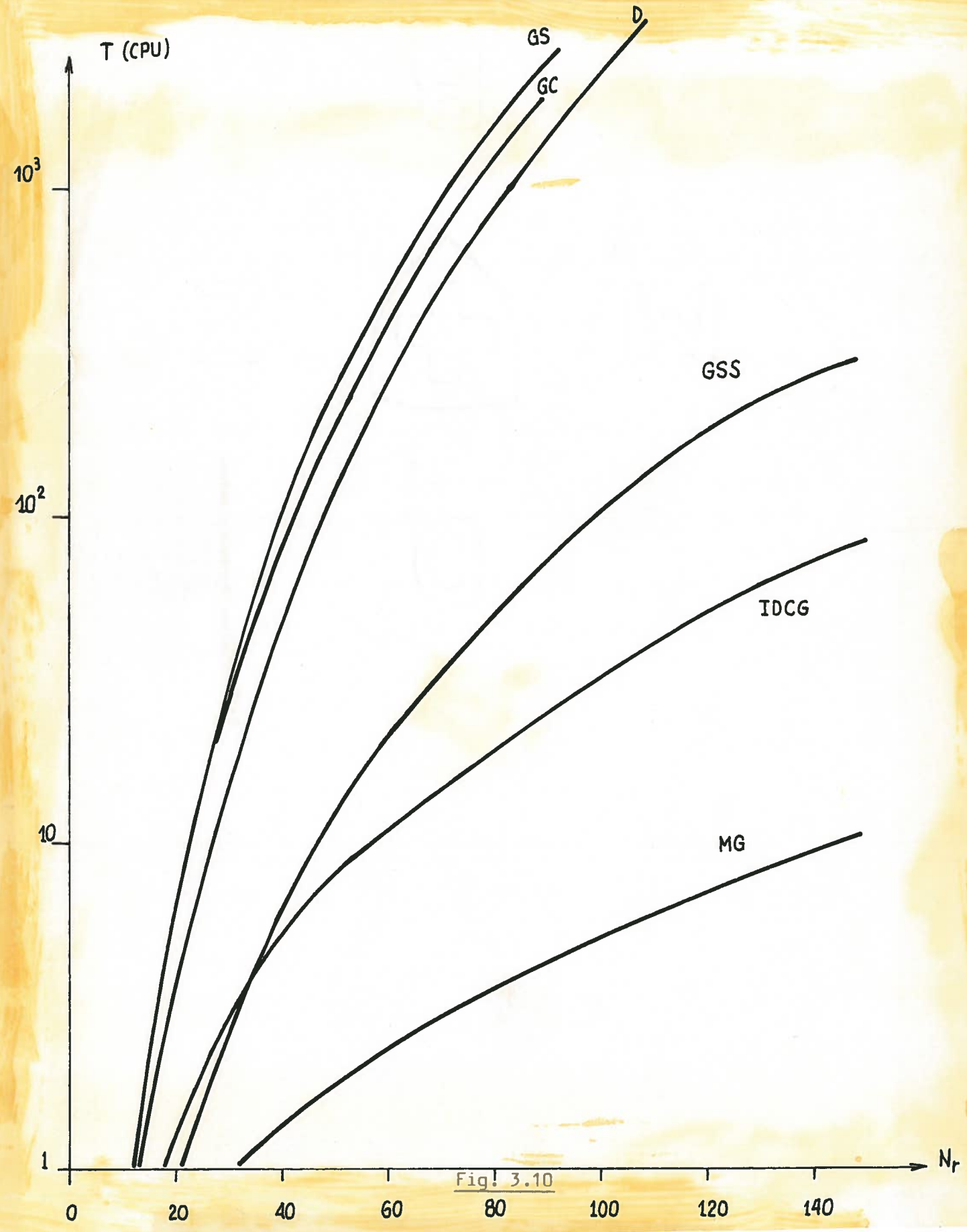


Fig. 3.10

3.5.6 COMPARAISON DES METHODES

Sur la figure 3.10, nous avons représenté le temps de calcul utilisé par les différentes méthodes décrites jusqu'ici, en fonction du nombre d'intervalles.

Pour une méthode directe, nous savons que le temps d'exécution croît proportionnellement à $N*M^2$, donc $\sim N_r^4$ si $N_z = N_r/2$. Cette estimation s'applique aussi aux méthodes de Gauss-Seidel et du gradient conjugué.

La surrelaxation avec le choix optimal de ω prend un temps d'exécution proportionnel à $N*M$, donc $\sim N_r^3$.

La méthode du gradient conjugué avec préconditionnement ainsi que la variante de la méthode à maillages multiples (MULTIGRID) n'exigent qu'un temps proportionnel à N, donc $\sim N_r^2$ et constituent les méthodes les plus efficaces.

Il faut encore noter que MULTIGRID demande une programmation méticuleuse si l'on veut bénéficier de tous ses avantages.

3.6 VALEURS PROPRES ET VECTEURS PROPRES: $A\underline{x} = \lambda I\underline{x}$

Le problème à valeurs propres

$$A\underline{x} = \lambda I\underline{x} \tag{3.57}$$

peut être multiplié à gauche par la transposée d'une matrice U conduisant à

$$U^T AU\underline{y} = \lambda U^T U\underline{y} \tag{3.58}$$

où $U\underline{y} = \underline{x}$.

Le problème (eq. 3.58) est identique au problème initial (eq. 3.57) si U est une matrice dite orthonormale satisfaisant à la condition

$$U^T U = I \tag{3.59}$$

En utilisant de telles matrices U, on peut transformer le problème (3.57) sans changer les valeurs propres. Cette idée est utilisée pour résoudre le problème (3.57) en deux pas:

(a) Suite de transformations orthogonales sur A afin d'arriver à une matrice tridiagonale T (transformations de Givens, Householder ou Lanczos).

(b) Calcul des valeurs propres de T par une méthode itérative comme la bissection ou un algorithme LR.

En dehors de ces méthodes ils en existent d'autres qui ne font appel qu'à (a) (méthode de Jacobi) ou qu'à (b) (itération inverse du vecteur).

3.6.1. Méthode de Jacobi

Les valeurs propres d'une matrice A ne changent pas lors d'une transformation orthogonale

$$\tilde{A} = U^T A U \tag{3.60}$$

où U satisfait à

$$U^T U = I$$

Pour la méthode de Jacobi, U s'écrit sous la forme:

$$U = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & \cos\theta & \sin\theta & \\ & & -\sin\theta & \cos\theta & \\ & & & & 1 \end{pmatrix} \begin{matrix} \\ \\ \leftarrow p \\ \leftarrow q \\ \\ \end{matrix} \qquad (3.61)$$

$$\begin{matrix} \\ \\ \uparrow \\ p \end{matrix} \qquad \begin{matrix} \\ \\ \uparrow \\ q \end{matrix} \qquad 1$$

où p = 1,..., N-1 et q = p+1,..., N cycliquement. La valeur de l'anqle de rotation θ pour p et q donnés est choisie de telle façon que l'élément de matrice transformé a_{pq} devienne zéro. Les éléments touchés sont ceux des lignes et colonnes p et q. La transformation s'écrit:

$$\begin{aligned}
\tilde{a}_{ij} &= a_{ij} && i,j \neq p \text{ ou } q \\
\tilde{a}_{jp} &= \tilde{a}_{pj} = a_{pj}\cos\theta - a_{qj}\sin\theta \\
\tilde{a}_{jq} &= \tilde{a}_{qj} = a_{pj}\sin\theta + a_{qj}\cos\theta && j \neq p,q \\
\tilde{a}_{pq} &= \tilde{a}_{pq} = 1/2(a_{pp} - a_{qq})\sin(2\theta) + a_{pq}\cos(2\theta) \\
\tilde{a}_{pp} &= a_{pp}\cos^2\theta - 2a_{pq}\sin\theta\cos\theta + a_{qq}\sin^2\theta \\
\tilde{a}_{qq} &= a_{qq}\sin^2\theta + 2a_{pq}\sin\theta\cos\theta + a_{qq}\cos^2\theta
\end{aligned} \qquad (3.62)$$

Avec ce processus, la somme de tous les éléments hors diagonale au carré va être diminuée par la quantité $2\,a_{pq}^2 - 2\,\tilde{a}_{pq}^2$. En demandant $\tilde{a}_{pq} = 0$, on réduit cette somme d'une façon optimale, donc

$$\tilde{a}_{pq} = 0 \rightarrow \operatorname{tg}(2\theta) = \frac{2a_{pq}}{a_{qq} - a_{pp}} \qquad (3.63)$$

Pour obtenir les vecteurs propres, on crée une matrice V qui, initialement égale l'identité, inclus toutes les transformations. Pour une rotation de Jacobi

$$\tilde{V} = U^{-1} V \equiv U^{T} V \tag{3.64}$$

ce qui correspond à effectuer les opérations

$$\left.\begin{aligned} \tilde{V}_{jp} &= V_{jp} \cos\theta - V_{jq} \sin\theta \\ \tilde{V}_{jq} &= V_{jp} \sin\theta + V_{jq} \cos\theta \end{aligned}\right\} \qquad j = 1, \ldots, n \tag{3.65}$$

Le nombre d'opérations par itération, lorsque l'on transforme tous les éléments de la matrice, est de l'ordre de $\sim 4N^3$. En plus, il faut noter que par cette méthode on remplit toute matrice initialement creuse. On a donc avantage à utiliser un autre processus.

3.6.2 Transformation de Householder

Au lieu d'essayer d'appliquer une suite infinie de transformations pour obtenir une matrice diagonale, on peut faire N-2 transformations orthogonales pour trouver une matrice tridiagonale T dont les valeurs propres peuvent être trouvées très efficacement. Choisissons un vecteur $\underline{w}$ de longueur $\underline{w}^T\underline{w}=1$. La matrice

$$H = I - 2\,\underline{w}\,\underline{w}^{T} \tag{3.66}$$

est une matrice orthogonale, puisque

$$H^{T}H = I - 4\,\underline{w}\,\underline{w}^{T} + 4\,\underline{w}\,\underbrace{\underline{w}^{T}\,\underline{w}}_{1}\,\underline{w}^{T} = I \tag{3.67}$$

L'idée est de trouver une suite de N-2 vecteurs $\underline{w}$ tels que les transformations

$$\tilde{A} = H^T A H = (I-2\,\underline{w}\,\underline{w}^T)\,A\,(I-2\,\underline{w}\,\underline{w}^T)$$

$$= A - 2\,\underline{w}\,\underline{w}^T A - 2\,A\,\underline{w}\,\underline{w}^T + 4\,\underline{w}\,\underline{w}^T A\,\underline{w}\,\underline{w}^T$$

$$= A - 2\,\underline{w}\,(A\underline{w} - Q(\underline{w})\underline{w})^T - 2(A\underline{w} - Q(\underline{w})\underline{w})\,\underline{w}^T$$

$$\tilde{A} = A - 2\,\underline{w}\,\underline{g}^T - 2\,\underline{g}\,\underline{w}^T \qquad (3.68)$$

où

$$\underline{g} = A\,\underline{w} - Q(\underline{w})\underline{w} = A\,\underline{w} - (A\,\underline{w},\,\underline{w})\,\underline{w} \qquad (3.69)$$

$Q(\underline{w}) = (A\underline{w},\underline{w})$

conduisent à une matrice tridiagonale T. Effectuons la première transformation: la matrice A initiale a la forme

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{12} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{N-1N} \\ a_{1N} & \cdots & a_{N-1N} & a_{NN} \end{pmatrix} \qquad (3.70)$$

et la matrice transformée $\tilde{A}$ après la première transformation

$$\tilde{A} = \begin{pmatrix} \tilde{a}_{11} & \cdots & \tilde{a}_{1\,N-1} & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \tilde{a}_{1\,N-1} & \ddots & \tilde{a}_{N-1N-1} & \tilde{a}_{N-1N} \\ 0 & \cdots & \tilde{a}_{N-1N} & \tilde{a}_{NN} \end{pmatrix} \qquad (3.71)$$

Par un bon choix de $\underline{w}$, on veut éliminer les éléments $\tilde{a}_{1\,N}$ à $\tilde{a}_{N-2\,N}$. Les nouveaux éléments de A sont:

$$\tilde{a}_{ij} = a_{ij} - 2\,w_i g_j - 2\,w_j g_i \qquad (3.72)$$

Pour $j = N$ et $w_N = 0$ (puisque nous avons un degré de liberté de trop, nous pouvons donc choisir une composante de $\underline{w}$).

$$\tilde{a}_{iN} = 0 = a_{iN} - 2 w_i g_N, \qquad i < N-1 \tag{3.73}$$

$$\tilde{a}_{N-1\ N} = a_{N-1\ N} - 2 w_{N-1} g_N \tag{3.74}$$

$$\tilde{a}_{N\ N} = a_{N\ N} \tag{3.75}$$

où

$$g_N = \sum_{i=1}^{N-1} a_{iN} w_i \tag{3.76}$$

Les équations (3.73) et (3.74) définissent les composantes de $\underline{w}$ comme fonction de g_N.

$$w_i = \frac{a_{iN}}{2g_N} \qquad i < N-1$$

$$w_{N-1} = \frac{a_{N-1N} - \tilde{a}_{N-1N}}{2\ g_N} \equiv \frac{a - b}{2\ g_N} \tag{3.77}$$

Puisque

$$\underline{w}^T \underline{w} = 1 = \sum_{i=1}^{N-2} \frac{a_{iN}^2}{4g_N^2} + \frac{a^2 - 2a\ b + b^2}{4g_N^2} \tag{3.78}$$

Nous obtenons

$$4g_N^2 = \sum_{i=1}^{N-1} a_{iN}^2 - 2\,ab + b^2 \qquad (3.79)$$

Remplaçons w_i (3.77) dans (3.76) et multiplions par $4g_N$, nous avons

$$4g_N^2 = 2\sum_{i=1}^{N-1} a_{iN}^2 - 2\,ab \qquad (3.80)$$

Identifier (3.79) et (3.80) conduit à

$$b^2 = \sum_{i=1}^{N-1} a_{iN}^2 . \qquad (3.81)$$

La transformation de Householder pour éliminer les éléments $a_{1\,N}$ à $a_{N-2\,N}$ de A se fait comme suit:

(a) $$\sigma = \sum_{i=1}^{N-1} a_{iN}^2$$

(b) $$b \equiv \tilde{a}_{N-1\,N} = \sqrt{\sigma}\ \frac{a_{N-1N}}{a_{N-1\,N}}$$

(c) $$g_n = \sqrt{\frac{\sigma - a_{N-1\,N}\,\tilde{a}_{N-1\,N}}{2}}$$

(d) $$w_i = \frac{a_{iN}}{2g_N} \qquad i = 1, \ldots, N-2$$

(e) $$w_{N-1} = \frac{a_{N-1\,N} - \tilde{a}_{N-1\,N}}{2g_N}$$

$$(f) \quad Z_k = (A\underline{w})_k = \sum_{j=1}^{N-1} a_{kj} w_j \qquad k = 1, \ldots, N-1$$

$$(g) \quad Q(\underline{w}) = (A\underline{w}, \underline{w}) = \sum_{k=1}^{N-1} w_k Z_k$$

$$(h) \quad g_k = Z_k - Q(\underline{w})\, w_k \qquad k = 1, \ldots, N-1$$

$$(i) \quad \tilde{a}_{ij} = a_{ij} - 2w_i g_j - 2w_j g_i \qquad \begin{matrix} i = 1, \ldots, N-1 \\ j = 1, \ldots, N-1 \end{matrix}$$

Le nombre d'opérations pour effectuer (f) et (i) est proportionnel à N^2. Si la matrice A est une matrice de bande de largeur M, toutes les w_i sont nuls sauf ceux pour $N-M/2 \leq i < N$. Le nombre d'opérations est alors proportionnel à M^2. Pour arriver à une matrice tridiagonale T, il faut faire N-2 transformations Householder ce qui donne un nombre d'opérations proportionnel à N^3 pour une matrice pleine et NM^2 pour une matrice de bande. Afin de pouvoir reconstituer les vecteurs propres, on doit garder toutes les vecteurs $\underline{w}$ de transformation. Les vecteur propres $\underline{y}$ du problème

$$T\underline{y} = \lambda I\underline{y} \tag{3.82}$$

sont après coup transformés cycliquement en

$$\underline{x} = (I-2\underline{w}\,\underline{w}^T)\underline{y} = \underline{y} - 2c\underline{w} \tag{3.83}$$

où $c = \underline{w}^T\underline{y}$, ce qui, pour toutes les transformations implique $\sim N^2$ opérations par vecteur propre pour une matrice pleine ou $\sim NM$ opérations par vecteur pour une matrice de bande.

3.6.3 Bissection

Cette méthode permettant de trouver une ou plusieurs valeurs propres d'une matrice tridiagonale, donc de

$$T\underline{y} = \lambda I\underline{y}, \tag{3.84}$$

est basée sur le théorème de Sylvestre.

Faisons d'abord un shift du spectre par λ_o,

$$\tilde{T}\underline{y} = \tilde{\lambda} I\underline{y} \tag{3.85}$$

où $\tilde{T} = T-\lambda_o I$ et $\tilde{\lambda} = \lambda - \lambda_o$.

Si l'on décompose la matrice symétrique $\tilde{T}$ en

$$\tilde{T} = LDL^T \tag{3.86}$$

qui constitue un processus de $\sim N$ opérations seulement, en appliquant le théorème de Sylvestre, le nombre de valeurs propres négatives de D est identique à celui de T. En variant λ_o on peut aisément savoir combien de valeurs propres se trouvent dans un domaine (voir figure 3.11). Le nombre de valeurs propres négatives de T correspond dans la matrice T au nombre de valeurs propres plus petites que λ_o. La méthode de la bissection localise une valeur propre λ en cherchant deux valeurs λ_1 et λ_2 ($\lambda_2 > \lambda_1$) du shift tel que $\lambda_1 < \lambda < \lambda_2$. Si n, n_1 et n_2 sont le nombre de valeurs propres plus petit que λ, λ_1 et λ_2, respectivement,

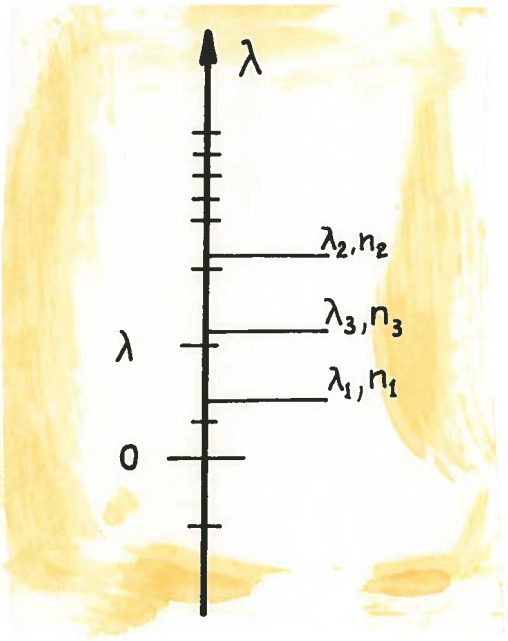


FIG. 3.11:
Spectre λ de T. n_1 et n_2 sont le nombre de valeurs propres plus petites que λ_1 et λ_2, respectivement. Dans le domaine $\lambda_1 < \lambda < \lambda_2$ il y a n2 - n1

$n_1 \leq n \leq n_2$ et $n_2 - n_1 \geq 1$. On peut se rapprocher d'une des valeurs propres du domaine en faisant comme prochain shift

$$\lambda_3 = \frac{\lambda_1 + \lambda_2}{2} \tag{3.87}$$

conduisant à un nouveau $n = n_3$. Si l'on est intéressé à la valeur propre numéro n, on fait des shifts jusqu'à ce qu'on trouve deux valeurs du shift λ_o, disons λ_k et λ_{k+1} tel que $n_k + 1 = n_{k+1}$ et $\lambda_k < \lambda < \lambda_{k+1}$. Après chaque pas d'une bissection additionnelle, donc la coupure du domaine spectrale, on diminue l'erreur de la valeur propre par un facteur deux. Cette méthode très sûre, évite de manquer une valeur propre dans le spectre. Une valeur propre multiple est obtenue avec la bonne multiplicité. Elle reste néanmoins puissante aussi longtemps que l'on ne s'intéresse qu'à un certain nombre de valeurs propres. Si on désire les avoir toutes, un autre algorithme (transformation LR, par exemple) devrait être utilisé.

3.6.4 Algorithmes LR et QD

La transformation LR permet de connaître, de façon plus efficace que par la méthode de la bissection, toutes les valeurs propres d'une matrice tridiagonale T qui doit être définie positive. D'abord, il faut donc rendre la matrice T définie positive grâce à un shift par un $\lambda_o > 0$ adéquat. On résoud donc

$$\tilde{T}\underline{y} = \tilde{\lambda} I \underline{y} \tag{3.88}$$

où $\tilde{\lambda} = \lambda + \lambda_o$ et $\tilde{T} = T + \lambda_o I$ est telle que la matrice diagonale D de la décomposition (eq. 3.86) n'aie plus de valeurs négatives. Une fois obtenue la matrice $\tilde{T}$ définie positive, on la décompose en

$$\tilde{T} = R_o^T R_o \tag{3.89}$$

et on crée une nouvelle matrice

$$\tilde{T}_o = R_o R^T \tag{3.90}$$

Cette nouvelle matrice $\tilde{T}_o$ est de nouveau décomposée en

$$\tilde{T}_o = R_1^T R_1 \qquad (3.91)$$

et la prochaine matrice $\tilde{T}_1$ sera formée par

$$\tilde{T}_1 = R_1 R_1^T \qquad (3.92)$$

Ce processus converge vers une matrice diagonale contenant toutes les valeurs propres. La convergence de cette méthode peut être très lente s'il y a des valeurs propres voisines. Pour améliorer la convergence, on élimine les valeurs propres déjà obtenues (on obtient d'abord les plus petites) et on déplace le spectre.

Afin de minimiser le nombre d'opérations ce processus s'appuie sur un algorithme QD (quotients-différences). La matrice tridiagonale symétrique définie positive

$$\tilde{T} = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_{N-1} \\ & & & \beta_{N-1} & \alpha_N \end{bmatrix} \qquad (3.93)$$

est d'abord transformée en une matrice non-symétrique

$$\hat{T} = \begin{bmatrix} \alpha_1 & 1 & & & \\ \beta_1^2 & \alpha_2 & 1 & & \\ & \beta_2^2 & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & \beta_{N-1}^2 & \alpha_N \end{bmatrix} \qquad (3.94)$$

telle que

$$\hat{T} = H^{-1} \tilde{T} H \qquad (3.95)$$

et

$$H = \begin{bmatrix} \beta_1 & & & \\ & \beta_2 & & \\ & & \ddots & \\ & & & \beta_N \end{bmatrix} \qquad (3.96)$$

La transformation (3.95) ne change pas les valeurs propres puisque $H^{-1}*H = I$. Par une suite de décompositions

$$\hat{T}_k = L_k R_k \tag{3.97}$$

où

$$L_k = \begin{bmatrix} 1 & & & \\ \ell_1 & 1 & & \\ & \ddots & \ddots & \\ & & \ell_{N-1} & 1 \end{bmatrix} \tag{3.98}$$

et

$$R_k = \begin{bmatrix} d_1 & 1 & & \\ & d_2 & \ddots & \\ & & \ddots & 1 \\ & & & d_N \end{bmatrix} \tag{3.99}$$

et de multiplications

$$\hat{T}_{k+1} = R_k L_k, \tag{3.100}$$

la matrice

$$\Lambda = \lim_{k \to \infty} \hat{T}_k \tag{3.101}$$

devient une matrice diagonale qui contient les valeurs propres.

Les deux opérations (3.97) et (3.100) sont très simple à faire. Pour passer de la matrice $\hat{T}$ (eq. 3.94) à L_k et R_k, il faut effectuer les opérations suivantes:

$$\left.\begin{aligned} d_1 &= \alpha_1 \\ \ell_i &= \beta_i^2/d_1 \\ \alpha_{i+1} &= \alpha_{i+1} - \ell_i \end{aligned}\right\} \; i = 1,\ldots, N-1 \qquad (3.102)$$
$$d_N = \alpha_N .$$

Pour obtenir $\hat{T}_{k+1}$, on multiplie R_k par L_k de la façon suivante:

$$\left.\begin{aligned} \alpha_i &= d_i + \ell_i \\ \beta_i^2 &= \ell_i * d_{i+1} \end{aligned}\right\} \; i = 1,\ldots, N-1 \qquad (3.103)$$
$$\alpha_N = d_N$$

3.7 Valeurs propres et vecteurs propres: $A\underline{x} = \lambda B\underline{x}$

3.7.1 Itération inverse du vecteur

Un problème à valeurs propres du type

$$A\underline{x} = \lambda B\underline{x} \qquad (3.104)$$

où A est symétrique et B définie positive, peut être directement abordé par la méthode de l'itération inverse du vecteur. Elle consiste à itérer de la façon suivante

$$A\underline{v}^{(k+1)} = B\underline{v}^{(k)}$$

et (3.105)

$$\underline{v}^{(k+1)} = \underline{v}^{(k+1)}/\|\underline{v}^{(k+1)}\|$$

Comme vecteur initial $\underline{v}^{(o)}$ on choisit en général un vecteur dont les composantes sont tirées au hasard. Le processus converge vers le vecteur propre qui correspond à la valeur propre la plus proche de zéro. Après convergence, la valeur propre est obtenue par la normalisation

$$\lambda^N = \text{sign}(\underline{v}^{(k)}/\underline{v}^{(k+1)})/\|\underline{v}^{(k+1)}\| \qquad (3.106)$$

ou par le quotient de Rayleigh

$$\lambda^R = \frac{(A\underline{v}^{(k+1)}, \underline{v}^{(k+1)})}{(B\underline{v}^{(k+1)}, \underline{v}^{(k+1)})} \tag{3.107}$$

Soient λ_1 et λ_2 les deux valeurs propres les plus proches de zéro $o < |\lambda_1| < |\lambda_2|$, chaque composante du vecteur propre approché $\underline{v}$ converge vers $\underline{x}$ suivant

$$(v_i^{(k+1)} - x_i)/(v_i^{(k)} - x_i) \approx \frac{\lambda_1}{\lambda_2} \tag{3.108}$$

et la valeur propre λ_1 calculée par le quotient de Rayleigh converge en

$$\frac{\lambda_1^{(k+1)} - \lambda_1}{\lambda_1^{(k)} - \lambda_1} \approx \left(\frac{\lambda_1}{\lambda_2}\right)^2 \tag{3.109}$$

Pour trouver n'importe quelle valeur propre du spectre on effectue auparavant un déplacement du spectre par λ_o, on résoud donc

$$\tilde{A}\underline{x} = \tilde{\lambda} B\underline{x}$$

où
$$\tilde{A} = A - \lambda_o B \tag{3.110}$$

et
$$\tilde{\lambda} = \lambda - \lambda_o$$

L'itération inverse (3.105) appliquée au problème (3.110) converge alors vers la valeur propre la plus proche de λ_o du problème initial (3.104). En variant λ_o, on peut, comme montré sur la figure 3.11, trouver chaque valeur et vecteur propre du spectre.

La résolution matricielle (3.105) s'effectue de la façon suivante:

(a) Décomposer $A = LDL^T$ (3.111)

(b) Résoudre itérativement $LDL^T \underline{x}^{(k+1)} = B\underline{x}^{(k)}$ (3.112)

La décomposition de A ne se fait qu'une seule fois, tandis que la résolution se fait itérativement.

Proposition: D a le même nombre de valeurs propres négatives que le problème (3.104).

Puisque B est définie positive, elle peut être décomposée en

$$B = R^T R \tag{3.113}$$

Multipliant (3.104) à gauche par R^{-T} et introduisant un nouveau vecteur

$$\underline{y} = R\underline{x} \tag{3.114}$$

le problème (3.104) s'écrit

$$\hat{A}\underline{y} = \lambda I\underline{y} \tag{3.115}$$

où

$$\hat{A} = R^{-T} A R^{-1} \tag{3.116}$$

En appliquant le théorème de Sylvestre à la transformation (3.116), la matrice A a le même nombre de valeurs propres que A ou D (voir équation 3.111).

3.7.2 Itération inverse simultanée des vecteurs

Dans un problème à valeurs propre ordinaire $A\underline{x} = \lambda I\underline{x}$, des vecteurs propres sont orthogonaux. Cela n'est plus le cas dans le problème généralisé (3.104). Mais nous avons vu que le problème (3.104) peut être ramené au problème ordinaire (3.115) par: - une décomposition dite de Choleski de B (3.113) - une multiplication de (3.105) à gauche par R^{-T} - une transformation du vecteur $\underline{x}$ en $\underline{y} = R\underline{x}$. Cela implique que les vecteurs propres $\underline{y}$ sont orthogonaux. Cette propriété peut être utilisée pour itérer simultanément sur m vecteurs. On procède comme suit:

(1) Shifter le spectre par λ_o.

(2) Décomposer $A = LDL^T$

(3) Décomposer $B = R^TR$

(4) Choisir vecteurs approchés $\underline{v}_i^{(o)}$, $i = 1, \ldots, m$

(5) $k = 0$

(6) $\underline{y}_i^{(k)} = R\,\underline{v}_i^{(k)}$

(7) Orthonormaliser $(\underline{y}_i^{(k)}, \underline{y}_j^{(k)}) = \delta_{ij}$, $i,j = 1, \ldots, m \rightarrow \lambda_i^N$

(8) Tests de convergence ok go to (14)

(9) $u_i^{(k)} = R^T\underline{y}_i^{(k)}$

(10) $LD\underline{w}_i^{(k+1)} = \underline{u}_i^{(k)}$

(11) $L^T\underline{v}_i^{(k+1)} = \underline{w}_i^{(k+1)}$

(12) $k = k + 1$

(13) go to (6)

(14) $\underline{x}_i = \underline{v}_i^{(k+1)}$

(15) Rayleigh Quotient $\lambda_i^R = \dfrac{(A\underline{x}_i, \underline{x}_i)}{(B\underline{x}_i, \underline{x}_i)}$

(16) Comparer λ_i^R avec λ_i^N

Pour $m > 1$, l'expérience montre qu'il est avantageux d'itérer simultanément sur le double de vecteurs propres, donc sur 2 m vecteurs si l'on en veut obtenir m.

3.8 BIBLIOGRAPHIE

H.R. Schwarz, H. Rutishauser, E. Stiefel "Numerik Symetrischer Matrizen" (1972, Teubner Verlag, en allemand) ou "Numerical Analysis of Symmetric Matrices" (1973, Prentice-Hall, en anglais) couvrant tout le domaine, sont très clairs et simples à lire.

La méthode du "Choleski Incomplet, Gradients Conjugués" est présentée dans D.S. Kershaw "The Incomplete Cholesky - Conjugate Gradient Method for the Iterative Solution of Systems of Linear Equations", Journal of Comp. Phys. 26, 43-65 (1978).

La méthode du MULTIGRID est décrite dans A. Brandt, "Multigrid Methods", (1982, Springer Verlag), ou "Multi-Level Adaptive Solutions to Boundary-Value problems", Math. Comp. 31, 333 - 390 (1977).

Deux livres sur les problèmes à valeurs propres sont à noter de J.H. Wilkinson "The Algebraic Eigenvalue Problem" (1965, Oxford Press) et de B.N. Partlett "The Symmetric Eigenvalue Problem" (1980, Prentice-Hall).

Les références concernant les librairies mathématiques sont:

J.J. Dongarra, C.B. Moler, J.R. Bunch and G.W. Stewart, "LINPACK Users' Guide", SIAM, Philadelphia, 1979.

Numerical Algorithms Group Library, Manuel NAG.

B.T. Smith et al. "Matrix Eigensystem Routines - EISPACK Guide", Lecture Notes in Computer Science 6 (1974, Springer Verlag), et

B.S. Garbow et al. "Matrix Eigensystem Routines - EISPACK Guide Extension", Lecture Notes in Computer Science 51 (1977, Springer Verlag).

Le programme permettant de trouver plusieurs valeurs et vecteurs propres du problème $A\underline{x} = \lambda B\underline{x}$ est décrit dans:

T. Ericsson "Spectral Transformation Lanczos Algorithm (STLM)" Manuel (1982), et

A. Ruhe and T. Ericsson "The Spectral Transformation Lanczos Method in the Numerical Solution of Large, Sparse, Generalized, Symmetric Eigenvalue Problems", Math. Comp. 35, 1251 - 1268 (1980).

Dans MATLIB se trouvent HYMNIA (pour des matrices A et B de bande) et HYMNIABLOCK (pour des matrices de blocs diagonaux A et B stockées sur disque) pour trouver un seul vecteur propre du problème $A\underline{x} = \lambda B\underline{x}$. Les déscriptions se trouvent dans

R. Gruber, "HYMNIA - Band Matrix Package for Solving Eigenvalue Problems", Comp. Phys. Comm. 10, 30 - 41 (1975), et

"HYMNIABLOCK - Eigenvalue Solver for Blocked Matrices", Comp. Phys. Comm. 20, 421 - 428 (1980).

Appendice 3A Système de programmation OLYMPUS

Ecrire et tester un programme d'ordinateur de taille moyenne (> 1000 cartes) ou grande (> 10000 cartes) est souvent très coûteux. Dépendant de la complexité du problème, on compte pour des problèmes scientifiques entre une et cinq instructions écrites par jour et personne. Ceci inclue la formulation du problème, le choix de la méthode numérique, l'organisation du code, la recherche de programmes existants dans les librairies mathématiques, l'écriture du code et les premières applications pour le tester. Pour une personne expérimentée, la partie écriture du code prend moins de 10% du temps total. Le choix du compilateur joue donc une importance minime. Les scientifiques choisissent en générale FORTRAN qui est un compilateur standardisé et très optimisé sur toutes les ordinateurs.

La partie organisation du code joue un rôle d'autant plus important que le code est grand et plus il y a des personnes qui doivent s'en occuper. Une méthode d'organisation de programmes de moyennes à grandes tailles est OLYMPUS proposée par K.V. Roberts. L'introduction de la première publication est donnée dans Fig. 3.12. Les idées principales d'OLYMPUS sont esquissées dans Fig. 3.13 avec Tables 2,3,5 et 6:

- Un programme d'ordinateur devrait être structuré. L'initialisation, la lecture des données, les calculs et la sortie des résultats sont bien séparés.

- Un programme d'ordinateur devrait être écrit d'une façon modulaire. Chaque tâche séparable devrait être traitée dans sa propre sousroutine. Les exemples sont donnés avec LABRUN, CLEAR, PRESET etc. qui sont expliqués dans le programme MUGRID de l'Appendice 3.C.

Dans OLYMPUS, les caractères initiaux des variables sont prescrits. Les variables qui entrent dans une sousroutine comme argument commencent, par exemple, par P ou K et les variables locales par Z ou I (voir Table 5 de Fig. 3.13).

Une bonne documentation d'un code est impérative si le programmeur aimerait encore comprendre son programme une année après ou s'il y a plusieurs auteurs. Un code d'ordinateur devrait être écrit d'une façon auto-explicative.

Tout cela est de la théorie, nous le savons, mais, si l'on s'efforce de suivre ces instructions, rapidement on en apercevra l'utilité.

COMPUTER PHYSICS COMMUNICATIONS 7 (1974) 237–243. © NORTH-HOLLAND PUBLISHING CO., AMSTERDAM

AN INTRODUCTION TO THE OLYMPUS SYSTEM

K.V. ROBERTS
UKAEA Research Group, Culham Laboratory, Abingdon, Berks., UK

Received 10 December 1973

A standard methodology has been established for the design, construction and operation of Fortran programs to solve initial-value and other problems. The following two papers describe the OLYMPUS library package which is used in implementing this methodology on the ICL 4/70 at Culham, together with an illustrative example of a laser fusion code called MEDUSA 1 which has been developed in this way. Subsequent papers will describe how OLYMPUS can be installed on IBM and CDC machines. This article provides a brief introduction to the OLYMPUS system and explains why it has been adopted.

Introduction

An earlier article [1] in this journal discussed the problem of publishing scientific Fortran programs. It suggested a number of general principles which, if followed, would make it easier to transfer such programs from one user or one computer system to another, and would enable them to be readily adapted to solve problems that were not envisaged at the time when they were first written.

These principles have now been incorporated into a programming system called OLYMPUS [2, 3], which is used by the Computational Physics Group at the UKAEA Culham Laboratory to support the design, construction and operation of all new Fortran programs. So far as possible all programs are built according to a common plan, which makes them more intelligible as well as speeding up most aspects of the programming process. The OLYMPUS system was originally developed for physical initial-value codes which solve partial-differential equations of the general form

$$\frac{\partial f}{\partial t} + G(f) = 0 \qquad (1)$$

subject to appropriate initial and boundary conditions, where f is the solution vector and G is a linear or nonlinear operator. Many interesting problems in hydrodynamics, plasma physics, astrophysics and other areas of classical physics can be expressed in this form, and it seemed worthwhile to establish a common programming strategy for dealing with them. Once the OLYMPUS system had been developed, however, it was found to be equally useful with only minor changes for a much larger class of computing work, including programs which have no connection with physics at all, for example the automatic documentation and flowcharting codes discussed in ref. [1].

OLYMPUS employs a standard set of files which are installed on-line in a system or user library. These files include a main program, subprograms, COMMON blocks, procedure files and so on. The following paper [2] describes the package which is used to build and test the system on the ICL 4/70 used at Culham, while refs. [4, 5] describe the corresponding packages which are used for the IBM 360/370 series and the CDC 6000/7000 series respectively. It is intended that OLYMPUS itself should organize most of the system-dependent features such as channel numbers, word and byte lengths, character codes etc., so that once it has been installed and tested on a new computer system the actual transfer of any program in the 'olympian' family is relatively straightforward. Ref. [6] in this issue describes one such program, MEDUSA 1, a 1-dimensional lagrangian laser fusion code. Other members of the family will be published in *Computer*

Fig. 3.12

Table 2
Program architecture.

INSTRUCTIONS	0	Control
	1	Prologue
	2	Calculation
Classes of subprograms	3	Output
	4	Epilogue
	5	Diagnostics
	U	Utilities
DATA	1	General Olympus data
	2	Physical problem
Groups of labelled COMMON blocks	3	Numerical scheme
	4	Housekeeping
	5	I/O and diagnostics

// SUBSTITUTE COMPHY. (2)

Table 6
Documentation techniques.

Titles and Headings
Comments
Indexes
Program commentary
References, equation numbers, cross-references
Program map

Automatic documentation

Clarifier
Flowcharter
Selective printing of given statement types

Table 3
Structure of the CRONUS dummy program.

Class / Subprogram	*Subprograms* 0	1	2	3	4	5
0	(MAIN)					
1	BASIC	LABRUN	STEPON	OUTPUT	TESEND	REPORT
2	MODIFY	CLEAR			ENDRUN	CLIST
3	COTROL	PRESET				ARRAYS
4	EXPERT	DATA				
5		AUXVAL				
6		INITAL				
7		RESUME				
8		START				

Common blocks
[Cl. 1] COMBAS Basic system parameters
[Cl. 9] COMDDP Development and diagnostic parameters

Table 5
Initial letters and array names

	Real, complex	Integer (and Hollerith)	Logical
Subprogram dummy arguments	P	K	KL
Common variable and array names	A–H, O, Q–Y	L, M, N	LL, ML, NL
Local variable and array names	Z	I	IL
Loop indexes		J	

Fig. 3.13

Appendice 3B Programme COURS

Dans ce programme, un système d'équations linéaires $A\underline{x} = \underline{b}$, où A et $\underline{b}$ se trouvent sur disque, est résolu par une élimination de Gauss en utilisant les librairies MATLIB, LINPACK et NAG et par différentes méthodes itératives, comme la méthode de Gauss-Seidel, sans et avec surrelaxation et la méthode du gradient conjugué sans et avec préconditionnement.

Nous y donnons le listing des routines de MATLIB utilisées ainsi que certaines sousroutines d'OLYMPUS et les routines de traitement de vecteurs SAXPY, SCOPY, SDOT, SSCAL, SXYPZ et VZERO provenant d'une CRAY 1.

```
*DECK MAIN
      PROGRAM COURS(OUTPUT,TAPE6=OUTPUT,TAPE10,TAPE12)
C
C
C     DANS CE PROGRAMME , UN SYSTEME D'EQUATIONS LINEAIRES
C
C                  A * X = B
C
C     ISSU DE LA DISCRETISATION PAR DIFFERENCES FINIES DE
C     L'EQUATION DE L'EQUILIBRE MAGNETOHYDRODYNAMIQUE EN GEO-
C     METRIE TORIQUE , EST RESOLU PAR DIFFERENTES METHODES :
C
C     METHODES DIRECTES  (ELIMINATION DE GAUSS)
C     *****************
C
C     -- BAND DE MATLIB
C     -- SGBFA ET SGBSL DE LINPACK
C     -- SGECO ET SGESL DE LINPACK
C     -- F01BMF ET F04AVF DE NAG
C
C
C     METHODES ITERATIVES
C     *******************
C
C     -- GAUSS - SEIDEL
C     -- GAUSS - SEIDEL AVEC SURRELAXATION
C     -- GRADIENTS CONJUGUES
C     -- GRADIENTS CONJUGUES AVEC PRECONDITIONNEMENT
C
C     LA METHODE A MAILLAGE MULTIPLES (MULTIGRID) EST PROGRAMMEE
C     DANS MUGRID PAR S.SEMENZATO .
C
C     A LA FIN DU PROGRAMME ,LES ROUTINES DE CALCUL DE
C     VALEURS PROPRES TRED1 ET TQL1 DE EISPACK SONT
C     UTILISEES POUR CALCULER TOUTES LES VALEURS PROPRES
C     DE LA MATRICE  ( A + AT ) / 2 ET DE LA MATRICE
C     PRECONDITIONNEE  ( L-1*A*R-1 + R-T*AT*L-T ) / 2 .
C
C
C     LES VARIABLES IMPORTANTES SONT :
C
C     N          LONGUEUR DE A
C     M=ML+MR+1  LARGEUR DE BANDE DE A
C     ND         RANGEES DIAGONALES NON ZEROS
C     ABND       MATRICE DE BANDE POUR MATLIB
C     ABLP       MATRICE DE BANDE POUR LINPACK
C     ANAG       MATRICE DE BANDE POUR NAG
C     AFULL      MATRICE PLEINE POUR LINPACK
C     ASPRS      ND RANGEES DIAGONALES POUR METHODES ITERATIVES
C     W          CONTIENT MEMBRE DE DROITE B
C     U          RESULTAT  X
C
C-----------------------------------------------------------------------
C
C
      DIMENSION ABND(3990),U(190),W(190)
      DIMENSION ABLP(31,190),IPVT(190)
      DIMENSION AFULL(190,190)
      DIMENSION ANAG(190,21),AL(190,10)
      DIMENSION ASPRS(190,5),ADEC(190,5),H(190),R(190),Y(190),WW(190)
```

```
      DIMENSION NLDIAG(2),NRDIAG(2)
      DIMENSION RR(19),RZ(10)
C
      COMMON AFULL
CC
C-----------------------------------------------------------------------
CL            0.        ANALYTIC SOLOVEV SOLUTION
C
      CALL PAGE
      NR=20
      NZ=10
      N=(NR-1)*NZ
      ASPCT=1.0/3.0
      ELLIPT=2.0
      PSIS=1.0/9.0
      RMAX=SQRT(1.+2.0*ASPCT)
      RMIN=SQRT(1.0-2.0*ASPCT)
      ZMIN=0.0
      ZMAX=ELLIPT*SQRT((1.0-SQRT(1.0-4.0*ASPCT**2))/2.0)
      DR=(RMAX-RMIN+2.0E-4)/NR
      DZ=(ZMAX-ZMIN+1.E-4)/NZ
C
CC    R AND Z GRID
C
      RR(1)=RMIN-1.E-4+DR
      NR1=NR-1
      DO  10  I=2,NR1
10    RR(I)=RR(I-1)+DR
      RZ(1)=0.0
      DO  20  I=2,NZ
20    RZ(I)=RZ(I-1)+DZ
C
CC    CALCULATE PSI(R,Z)
C
      CALL VZERO(W,N)
      K=0
      DO  30  I=1,NR1
      ZR=RR(I)
      DO  30  J=1,NZ
      K=K+1
      ZZ=RZ(J)
      W(K)=ASPCT**2*((ZR*ZZ/ELLIPT)**2+(ZR**2-1.0)**2/4.0)/PSIS
      W(K)=W(K)-PSIS
      IF (W(K) .GT. 0.0) W(K)=0.0
30    CONTINUE
      CALL TITLE(40HSOLUTION ANALYTIQUE DE SOLOVEV          )
      CALL RARRAY(8HXANALYT=,W,N)
C
C-----------------------------------------------------------------------
CL            1.        FILL IN ABND,ASPRS,AELP,AFULL,ANAG
CC
CC    READ IN BANDED MATRIX
C
      READ (12) NR,NZ,NA,N,(ABND(I),I=1,NA),(U(I),I=1,N)
      CALL SSCAL(NA,-1.0,ABND,1)
      CALL SSCAL(N,-1.0,U,1)
      M=NA/N
      ND=5
      ML=NZ
```

```
      MR=NZ
      EPS=1.E-6
      EPSMAC=1.E-14
C
CC    COPY ON SPARSE MATRIX
C
      CALL SCOPY(N,ABND(1),M,ASPRS(1,1),1)
      CALL SCOPY(N,ABND(ML),M,ASPRS(1,2),1)
      CALL SCOPY(N,ABND(ML+1),M,ASPRS(1,3),1)
      CALL SCOPY(N,ABND(ML+2),M,ASPRS(1,4),1)
      CALL SCOPY(N,ABND(M),M,ASPRS(1,5),1)
      CALL SCOPY(N,U,1,W,1)
C
CC    FILL BAND MATRIX FOR LINPACK
C
      NLP=N*(M+ML)
      CALL VZERO(ABLP,NLP)
      CALL SCOPY(N-ML,ASPRS(ML+1,1),1,ABLP(ML+M,1),ML+M)
      CALL SCOPY(N-1,ASPRS(2,2),1,ABLP(M+1,1),ML+M)
      CALL SCOPY(N,ASPRS(1,3),1,ABLP(M,1),ML+M)
      CALL SCOPY(N-1,ASPRS(1,4),1,ABLP(M-1,2),ML+M)
      CALL SCOPY(N-MR,ASPRS(1,5),1,ABLP(ML+1,MR+1),ML+M)
C
CC    FILL FULL MATRIX FOR LINPACK
C
      CALL VZERO(AFULL,N*N)
      CALL SCOPY(N-ML,ASPRS(ML+1,1),1,AFULL(ML+1,1),N+1)
      CALL SCOPY(N-1,ASPRS(2,2),1,AFULL(2,1),N+1)
      CALL SCOPY(N,ASPRS(1,3),1,AFULL(1,1),N+1)
      CALL SCOPY(N-1,ASPRS(1,4),1,AFULL(1,2),N+1)
      CALL SCOPY(N-MR,ASPRS(1,5),1,AFULL(1,MR+1),N+1)
C
CC    FILL BAND MATRIX FOR NAG LIBRARY
C
      CALL VZERO(ANAG,NA)
      IBND=ML+1
      DO  210  J=1,ML
      CALL SCOPY(MR+J,ABND(IBND),1,ANAG(J,1),N)
      IBND=IBND+M-1
 210  CONTINUE
      IBND=ML*M+1
      IML=ML+1
      DO  220  J=IML,N
      CALL SCOPY(M,ABND(IBND),1,ANAG(J,1),N)
      IBND=IBND+M
 220  CONTINUE
C
C-----------------------------------------------------------------------
CL            2.        SOLVE SYSTEM WITH DIFFERENT LIBRARIES
C
CC    SOLVE SYSTEM WITH BANDED MATRIX
C
      CALL TITLE(40HDECOMPOSE A=LDR AVEC BALOR DE MATLIB    )
      CALL CPUTIM
      CALL BAND(ABND,U,EPSMAC,N,ML,MR,NSING)
      CALL TITLE(40HRESOUD LDR*X=B AVEC BAND DE MATLIB      )
      CALL CPUTIM
      IF (NSING .EQ. 0) GO TO 230
      CALLMESAGE(48H *   MATRICE  A  EST  SINGULIERE   *           )
      STOP
```

```
 230      CONTINUE
          CALL RARRAY(8HX =      ,U,N)
C
C     SOLVE BANDED SYSTEM WITH LINPACK
C
          CALL SCOPY(N,W,1,U,1)
          CALL TITLE(40HDECOMPOSE A=LDR AVEC SGBFA DE LINPACK   )
          CALL CPUTIM
          CALL SGBFA(ABLP,M+ML,N,ML,MR,IPVT,INFO)
          CALL CPUTIM
          CALL TITLE(40HRESOUD LDR*X=B AVEC SGBSL DE LINPACK    )
          CALL IARRAY (8HIPIVOT =,IPVT,N)
          CALL CPUTIM
          CALL SGBSL(ABLP,M+ML,N,ML,MR,IPVT,U,0)
          CALL CPUTIM
          CALL RARRAY(8HX =      ,U,N)
C
C     SOLVE FULL SYSTEM WITH LINPACK
C
          REWIND 10
          WRITE (10) AFULL
          CALL SCOPY(N,W,1,U,1)
          CALL TITLE(40HDECOMPOSE MATRICE PLEINE AVEC SGECO     )
          CALL CPUTIM
          CALL SGECO(AFULL,N,N,IPVT,RCOND,H)
          CALL CPUTIM
          CALL RVAR(8HCOND(A)=,RCOND)
          CALL TITLE(40HRESOUD MATRICE PLEINE AVEC SGESL        )
          CALL CPUTIM
          CALL SGESL(AFULL,N,N,IPVT,U,0)
          CALL CPUTIM
          CALL RARRAY(8HX =      ,U,N)
C
C     SOLVE BANDED SYSTEM WITH NAG
C
          CALL TITLE(40HDECOMPOSE A=LDR AVEC F01BMF DE NAG      )
          CALL CPUTIM
          CALL F01BMF(N,ML,MR,ANAG,N,M,AL,N,ML,IPVT,IV,IFAIL)
          CALL CPUTIM
          IF (IFAIL .EQ. 0) GO TO 240
          CALLMESAGE(48H *   MATRICE  A  EST  SINGULIERE   *          )
          STOP
 240      CONTINUE
          CALL SCOPY(N,W,1,U,1)
          CALL TITLE(40HRESOUD LDR*X=B AVEC F04AVF DE NAG       )
          CALL CPUTIM
          CALL F04AVF(N,ML,MR,1,ANAG,N,M,AL,N,ML,IPVT,U,N)
          CALL CPUTIM
          CALL RARRAY(8HX =      ,U,N)
C
C-----------------------------------------------------------------
CL              3.        SOLVE SYSTEM ITERATIVELY
C
C
C     GAUSS - SEIDEL
C
          CALL TITLE(40HGAUSS - SEIDEL                          )
          CALL SCOPY(ND*N,ASPRS(1,1),1,ADEC(1,1),1)
          CALL CPUTIM
          OMEGA=1.0
```

```
      CALL GS(N,ND,ML,MR,ADEC,W,U,Y,R,OMEGA,EPS,ITER)
      CALL CPUTIM
      CALL IVAR (8HITER =  ,ITER)
      CALL RARRAY(8HX   =    ,U,N)
C
CC    GAUSS - SEIDEL WITH OVERRELAXATION
C
      OMEGA=1.77
      CALL TITLE(40HGAUSS - SEIDEL AVEC SURRELAXATION        )
      CALL SCOPY(ND*N,ASPRS(1,1),1,ADEC(1,1),1)
      CALL CPUTIM
      CALL GS(N,ND,ML,MR,ADEC,W,U,Y,R,OMEGA,EPS,ITER)
      CALL CPUTIM
      CALL RVAR(8HOMEGA = ,OMEGA)
      CALL IVAR (8HITER =  ,ITER)
      CALL RARRAY(8HX   =    ,U,N)
C
CC    CONJUGENT GRADIENTS
C
      CALL SCOPY(N,W,1,U,1)
      NLDIAG(1)=ML+1
      NLDIAG(2)=2
      NRDIAG(1)=2
      NRDIAG(2)=MR+1
      MR=2
      ML=2
      CALL SCOPY(N,U,1,Y,1)
      CALL TITLE(40HGRADIENTS CONJUGUES                      )
      CALL CPUTIM
      IC=0
      CALL SLIDCG(IC,ASPRS,ADEC,H,R,U,WW,Y,N,ND,ML,NLDIAG,MR,NRDIAG)
      CALL CPUTIM
      CALL IVAR(8HITER =  ,IC)
      CALL RARRAY(8HX   =    ,Y,N)
C
CC    CONJUGENT GRADIENTS WITH PRECONDITIONNING
C
      CALL SCOPY(N,W,1,U,1)
      CALL SCOPY(N,U,1,Y,1)
      CALL TITLE(40HDECOMPOSITION INCOMPLETE ET CG           )
      CALL CPUTIM
      IC=1
      CALL SLIDCG(IC,ASPRS,ADEC,H,R,U,WW,Y,N,ND,ML,NLDIAG,MR,NRDIAG)
      CALL CPUTIM
      CALL IVAR(8HITER =  ,IC)
      CALL RARRAY(8HX =    ,Y,N)
C
CC    ALL EIGENVALUES OF  L-1 * A * R-1  SYMETRIZED
C
      REWIND 10
      READ (10) AFULL
      CALL TITLE(40HVALEURS PROPRES DE  L-1 * A * R-1  SYM. )
      DO  310  I=1,N
      CALL SLWU(ADEC,AFULL(1,I),N,ND,ML,NLDIAG,MR,NRDIAG)
310   CONTINUE
      DO  320  I=1,N
      IN=N-NRDIAG(2)-I+2
      IA=I+NRDIAG(2)-1
      CALL SSCAL(N,1./ADEC(I,3),AFULL(1,I),1)
      IF (I .EQ. N) GO TO 320
```

```
      CALL SAXPY(N,-ADEC(I,4),AFULL(1,I),1,AFULL(1,I+1),1)
      IF (IN .LT. 1) GO TO 320
      CALL SAXPY(N,-ADEC(I,5),AFULL(1,I),1,AFULL(1,IA),1)
320   CONTINUE
      CALL SYMETR(AFULL,H,N,N)
      CALL CPUTIM
      CALL TRED1(N,N,AFULL,H,R,R)
      CALL TQL1(N,H,R,IERR)
      IF (IERR .NE. 0) STOP
      CALL CPUTIM
      CALL RARRAY(8HLAMBDA =,H,N)
C
C     ALL EIGENVALUES OF  A  SYMETRIZED
C
      REWIND 10
      READ (10) AFULL
      CALL TITLE(40HVALEURS PROPRES DE  A  SYMETRISEE        )
      CALL SYMETR(AFULL,H,N,N)
      CALL TRED1(N,N,AFULL,H,R,R)
      CALL TQL1(N,H,R,IERR)
      CALL RARRAY(8HLAMBDA =,H,N)
C
      STOP
      END
LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN MNS01
*DECK MNS01
      SUBROUTINE BAND(A,U,EPS,N,ML,MR,NSING)
C
C  MNS01 SOLVES BANDED SYSTEM OF LINEAR EQUATIONS
C
      DIMENSION A(1),U(1)
C
C     A IS A BAND MATRIX WITH ML LOWER DIAGONALS AND MR UPPER
C     DIAGONALS AND LENGTH N. L AND R CONTAIN 1 IN DIAGONAL WHICH
C     ARE NOT STORED. AFTER DECOMPOSITION L IS IN LOWER DIAGONAL
C     PART OF A, D IN DIAGONAL AND R IN UPPER DIAGONAL PART OF A.
C     NSING=-1 , IF DOMINANT SUBDETERMINANT OF A IS SINGULAR.
C
C     DECOMPOSE A
C
      CALL BALDR(A,EPS,N,ML,MR,NSING)
      CALL CPUTIM
C
C     SOLVE L * D * W = U
C
      CALL BLDWU(A,U,N,ML,MR)
C
C     SOLVE R * X = W
C
      CALL BRXW(A,U,N,ML,MR)
C
      RETURN
      END
LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN MNS02
*DECK MNS02
      SUBROUTINE BALDR(A,EPS,N,ML,MR,NSING)
```

```
C
C   MNS02 DECOMPOSE A=L*D*R
C
C       A IS A BAND MATRIX WITH ML LOWER DIAGONALS AND MR UPPER
C       DIAGONALS AND LENGTH N. L AND R CONTAIN 1 IN DIAGONAL WHICH
C       ARE NOT STORED. AFTER DECOMPOSITION L IS IN LOWER DIAGONAL
C       PART OF A, D IN DIAGONAL AND R IN UPPER DIAGONAL PART OF A.
C       NSING=-1 , IF DOMINANT SUBDETERMINANT OF A IS SINGULAR.
C
          DIMENSION A(1)
C
C       FILL UPPER TRIANGLE OF A WITH ZEROS
C
          M=ML+MR+1
          IU=1
          DO  10  IZ=1,ML
          CALL VZERO(A(IU),ML-IZ+1)
          IU=IU+M
   10     CONTINUE
C
C       INITIALIZE
C
          NSING=-1
          IKD=ML+1
          ZAD=ABS(A(IKD))*EPS
          IF (ZAD .EQ. 0) RETURN
C
C       SCAN OVER WHOLE LENGTH OF A
C
          DO  30  JN=2,N
C
C       TEST FOR ZERO PIVOT
C
          IF (ABS(A(IKD)) .LT. ZAD) RETURN
          IDNEXT=IKD+M
          ZAD=ABS(A(IDNEXT))*EPS
C
C       DIVIDE ROW BY PIVOT
C
          ZPIVOT=1.0/A(IKD)
          CALL SSCAL(MR,ZPIVOT,A(IKD+1),1)
C
C       SCAN OVER ALL ROWS FOR RECTANGULAR RULE
C
          ILEFT=IKD
          IL=ML
          IF (N-JN+1 .LT. ML) IL=N-JN+1
          IROW=MR+IL-ML
          DO  20  JROW=1,IL
          ILEFT=ILEFT+M-1
          ZLEFT=A(ILEFT)
          IF (ZLEFT .EQ. 0.0) GO TO 20
          A(ILEFT)=A(ILEFT)*ZPIVOT
          CALL SAXPY(IROW,-ZLEFT,A(IKD+1),1,A(ILEFT+1),1)
   20     CONTINUE
          IKD=IKD+M
   30     CONTINUE
          IKD=N*M-MR
          IF (ABS(A(IKD)) .LT. ZAD) RETURN
C
```

```
C      DECOMPOSITION O.K.
C
          NSING=0
C
C      FILL LOWER TRIANGLE OF A WITH ZEROS
C
          IL=IKD+1
          DO  40  IZ=1,MR
          CALL VZERO(A(IL),MR-IZ+1)
          IL=IL-M+1
   40     CONTINUE
C
          RETURN
          END
```

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN MNS03

```
*DECK MNS03
          SUBROUTINE BLDWU(A,U,N,ML,MR)
C
C   MNS03  SOLVE L*D*W=U
C
C      L IS THE LOWER DIAGONAL PART OF A. D IS DIAGONAL OF A.
C      THE VECTOR U BECOMES W AT OUTPUT.
C
          DIMENSION A(1),U(1)
C
C      FIRST N-ML COMPONENTS OF L*V=U
C
          M=ML+MR+1
          MD=ML
          NM=N-ML
          DO  100  I=1,NM
          MD=MD+M
          CALL SAXPY(ML,-U(I),A(MD),M-1,U(I+1),1)
C
C      DIVIDE BY DIAGONAL
C
          U(I)=U(I)/A(MD-M+1)
  100     CONTINUE
C
C      LAST ML-1 COMPONENTS OF L*V=U
C
          IF (ML .EQ. 1) GO TO 120
          DO  110  I=2,ML
          MD=MD+M
          CALL SAXPY(ML-I+1,-U(NM+I-1),A(MD),M-1,U(NM+I),1)
C
C      DIVIDE BY DIAGONAL
C
          U(NM+I-1)=U(NM+I-1)/A(MD-M+1)
  110     CONTINUE
  120     CONTINUE
C
C      DIVIDE BY LAST DIAGONAL
C
          U(N)=U(N)/A(N*M-MR)
C
          RETURN
          END
```

```
*DECK MNS04
      SUBROUTINE BRXW(A,U,N,ML,MR)
C
C  MNS04  SOLVE R*X=W
C
C     R IS THE UPPER DIAGONAL PART OF A.
C     THE VECTOR U IS W AT INPUT AND X AT OUTPUT.
C
      DIMENSION A(1),U(1)
C
C     LOWER N-MR COMPONENTS OF X
C
      M=ML+MR+1
      MD=M*(N-MR)
      NR=N-MR
      DO  100  I=1,NR
      CALL SAXPY(MR,-U(N-I+1),A(MD),M-1,U(NR-I+1),1)
      MD=MD-M
  100 CONTINUE
C
C     FIRST MR-1 COMPONENTS OF X
C
      IF (MR .EQ. 1) GO TO 120
      DO  110  I=2,MR
      CALL SAXPY(MR-I+1,-U(MR-I+2),A(M-I+1),M-1,U(1),1)
  110 CONTINUE
  120 CONTINUE
C
      RETURN
      END
LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN MNS05
*DECK MNS05
      SUBROUTINE BUAX(A,X,U,N,ML,MR)
C
C  MNS05  MULTIPLY U = A * X
C
C     B IS A BAND MATRIX OF LENGTH N AND BAND WIDTH ML+MR+1.
C     ML IS THE NUMBER OF LOWER OFF DIAGONALS AND MR THE NUMBER OF
C     UPPER OFF DIAGONALS. U AND X ARE VECTORS OF LENGTH N.
C
      DIMENSION A(1),X(1),U(1)
C
      CALL VZERO(U,N)
      M=ML+MR+1
C
C     THE ML LEFT OFF DIAGONALS
C
      IB=M*ML+1
      DO  100  I=1,ML
      CALL SXYPZ(N-ML-1+I,A(IB),M,X(1),1,U(ML+2-I),1)
      IB=IB-M+1
  100 CONTINUE
C
C     THE DIAGONAL
C
      CALL SXYPZ(N,A(ML+1),M,X(1),1,U(1),1)
C
C     THE MR RIGHT OFF DIAGONALS
```

```
C
      DO  110  I=1,MR
      CALL SXYPZ(N-I,A(ML+I+1),M,X(I+1),1,U(1),1)
 110  CONTINUE
C
      RETURN
      END
```

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN MNF01

```
*DECK MNF01
      SUBROUTINE FATA(A,Z,N)
C
C     MULTIPLIES  A = AT * A
C
      DIMENSION A(N,N),Z(N)
      DO  110  J=1,N
      CALL VZERO(Z,N)
      DO  100  I=J,N
      Z(I)=SDOT(N,A(1,J),1,A(1,I),1)
 100  CONTINUE
      CALL SCOPY(N-J+1,Z(J),1,A(J,J),1)
 110  CONTINUE
      DO  120  J=2,N
      DO  120  I=J,N
      A(J,I)=A(I,J)
 120  CONTINUE
      RETURN
      END
```

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN MNF02

```
*DECK MNF02
      SUBROUTINE SYMETR(A,Z,N,ND)
C
C     SYMETRIZES FULL MATRIX A
C
      DIMENSION A(ND,ND),Z(N)
      DO  110  J=1,N
      CALL SCOPY(N-J+1,A(J,J),1,Z(J),1)
      CALL SAXPY(N-J+1,1.0,A(J,J),ND,Z(J),1)
      CALL SSCAL(N-J+1,0.5,Z(J),1)
      CALL SCOPY(N-J+1,Z(J),1,A(J,J),1)
      CALL SCOPY(N-J+1,Z(J),1,A(J,J),ND)
 110  CONTINUE
C
      RETURN
      END
```

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN GSOR01

```
*DECK GSOR01
      SUBROUTINE GS(N,ND,ML,MR,A,B,X,Y,Z,OMEGA,EPS,ITER)
C
C     GAUSS - SEIDEL WITH OVERRELAXATION
C
      DIMENSION A(N,ND),B(N),X(N),Y(N),Z(N)
C
      SGS2=1.E4
      ITER=0
```

```
      CALL VZERO(X,N)
100   CONTINUE
      ITER=ITER+1
      SGS1=SGS2
      CALL SCOPY(N,B,1,Y,1)
      CALL SCOPY(N,X,1,Z,1)
C
CC    FORM Y=B-F*X
C
      CALL SSCAL(N,-1.,Z,1)
      CALL SXYPZ(N-1,A(1,4),1,Z(2),1,Y(1),1)
      CALL SXYPZ(N-MR,A(1,5),1,Z(MR+1),1,Y(1),1)
      CALL SCOPY(N,X,1,Z,1)
C
CC    SWEEP THE MESH
C
      X(1)=Y(1)/A(1,3)
      X(1)=OMEGA*X(1)+(1.-OMEGA)*Z(1)
      DO 110 I=2,ML
      X(I)=(Y(I)-X(I-1)*A(I,2))/A(I,3)
110   X(I)=OMEGA*X(I)+(1.-OMEGA)*Z(I)
      ML1=ML+1
      DO 120 I=ML1,N
      X(I)=(Y(I)-X(I-1)*A(I,2)-X(I-ML)*A(I,1))/A(I,3)
120   X(I)=OMEGA*X(I)+(1.-OMEGA)*Z(I)
C
CC    CHECK DIFFERENCE OF NORM
C
      SGS2=0.0
      DO 130 I=1,N
130   SGS2=SGS2+X(I)**2
      IF (ITER .GT. 1000) GO TO 100
      IF (ABS(SGS2-SGS1)/(SGS1+SGS2) .GT. EPS) GOTO 100
C
      RETURN
      END
```

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN IDCG01

```
*DECK IDCG01
      SUBROUTINE SLIDCG(IC,A,ADEC,H,R,U,W,Y,N,ND,ML,NLDIAG,MR,NRDIAG)
C
CC    SOLVE SPARSE SYSTEM OF LINEAR EQUATIONS
C
      DIMENSION A(N,ND),ADEC(N,ND),H(N),U(N),W(N),Y(N),R(N),
     D          NLDIAG(ML),NRDIAG(MR)
C
CC    U IS RIGHT HAND SIDE
C
CC    FILL UPPER TRIANGLE WITH ZEROS
C
      DO 10 I=1,ML
      II=NLDIAG(I)-1
      DO 10 J=1,II
      A(J,I)=0.
10    CONTINUE
C
CC    FILL LOWER TRIANGLE WITH ZEROS
C
      DO 20 I=1,MR
```

```
            II=NRDIAG(I)-1
            DO  20  J=1,II
            JJ=N-J+1
            A(JJ,ML+I+1)=0.
   20       CONTINUE
C
C       INITIALIZATION
C
            CALL VZERO(P,N)
            CALL VZERO(R,N)
            CALL VZERO(W,N)
            IF (IC .EQ. 0) GO TO 200
            CALL SCOPY(N*ND,A,1,ADEC,1)
C
C       INCOMPLETE DECOMPOSITION OF A , RESULT IN ADEC
C
            CALL SAILR(ADEC,N,ND,ML,NLDIAG,MR,NRDIAG)
C
C       NEW RIGHT HAND SIDE VECTOR
C
            CALL ATILDA(2,A,ADEC,U,Y,N,ND,ML,NLDIAG,MR,NRDIAG)
C
C       INITIALIZE CONJUGENT GRADIENT ITERATION
C
            CALL SCOPY(N,Y,1,U,1)
   200      CONTINUE
            CALL VZERO(Y,N)
            CALL SCOPY(N,U,1,R,1)
            CALL SAXPY(N,-1.,W,1,R,1)
            CALL SCOPY(N,R,1,H,1)
            RI=SDOT(N,R,1,R,1)
            R0=RI
C
C       CONJUGENT GRADIENT ITERATION
C
            DO  330  IT=1,500
            CALL SCOPY(N,H,1,W,1)
            IF (IC .EQ. 1) GO TO 310
            CALL SUAX(A,W,U,N,ND,ML,NLDIAG,MR,NRDIAG)
            CALL SCOPY(N,U,1,W,1)
            GO TO 320
   310      CONTINUE
            CALL ATILDA(1,A,ADEC,U,W,N,ND,ML,NLDIAG,MR,NRDIAG)
   320      CONTINUE
            RIM=RI
            ALM=RIM/SDOT(N,H,1,W,1)
            CALL SAXPY(N,ALM,H,1,Y,1)
            CALL SCOPY(N,R,1,U,1)
            CALL SAXPY(N,-ALM,W,1,R,1)
            RIJ=SDOT(N,R,1,U,1)
            RI=SDOT(N,R,1,R,1)
            EPM=RI/RIM
            CALL SSCAL(N,EPM,H,1)
            CALL SAXPY(N,1.,R,1,H,1)
            IF (SQRT(RI/R0) .LT. 1.E-6) GO TO 400
   330      CONTINUE
C
C       SOLVE R * Y = Y (SOLUTION VECTOR X)
C
   400      CONTINUE
```

```
      IF (IC .EQ. 1) CALL SRXW(ADEC,Y,N,ND,ML,NLDIAG,MR,NRDIAG)
      IC=IT
C
      RETURN
      END
```

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN IDCG02

```
*DECK IDCG02
      SUBROUTINE SAILR(A,N,ND,ML,NLDIAG,MR,NRDIAG)
C
C     DECOMPOSE A APPROXIMATELY INTO L * R
C
C*****                                  *****
C*****   ONLY FOR MATRICES WITH         *****
C*****    1 , 3 , 1  STRUCTURE          *****
C*****                                  *****
C
      DIMENSION A(N,ND),NLDIAG(ML),NRDIAG(MR)
C
C     A(IN)    SPARSE MATRIX TO BE INCOMPLETELY DECOMPOSED
C     A(OUT)   MATRIX R IN DIAGONAL AND UPPER DIAGONAL PART
C              MATRIX L IN LOWER DIAGONAL PART . UNITIES ON DIAGONAL NOT STORED
C     N        NUMBER OF UNKNOWNS
C     ND       =ML+MR+1
C     ML       NUMBER OF NONZERO LOWER DIAGONALS
C     NLDIAG   VECTOR POSITION OF LOWER DIAGONALS
C     MR       NUMBER OF NONZERO UPPER DIAGONALS
C     NRDIAG   VECTOR POSITION OF UPPER DIAGONALS
C
C     PRODUCE ALL DIAGONALS OF R
C
      JSTART=NLDIAG(1)-2
C
C     HERE NLDIAG=NRDIAG
C
      DO  100  I=2,N
      IS=JSTART+I
      A(I,3)=A(I,3)-A(I,2)*A(I-1,4)/A(I-1,3)
      IF (IS .LE. N) A(IS,3)=A(IS,3)-A(I-1,5)*A(IS,1)/A(I-1,3)
      DIAG=1.0
      IF (A(I-1,3) .GT. 0.0) DIAG=SQRT(A(I-1,3))
      A(I-1,3)=DIAG
      A(I-1,4)=A(I-1,4)/DIAG
      A(I-1,5)=A(I-1,5)/DIAG
      A(I,2)=A(I,2)/DIAG
      IF (IS .GT. N) GO TO 100
      A(IS,1)=A(IS,1)/DIAG
 100  CONTINUE
      DIAG=1.0
      IF (A(N,3) .GT. 0.0) DIAG=SQRT(A(N,3))
      A(N,3)=DIAG
C
      RETURN
      END
```

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN IDCG03

```
*DECK IDCG03
      SUBROUTINE ATILDA(K,A,ADEC,U,Y,N,ND,ML,NLDIAG,MR,NRDIAG)
```

```
C
CC      CALCULATE ATILDA*Y
C
        DIMENSION A(N,ND),ADEC(N,ND),U(N),Y(N),NLDIAG(ML),NRDIAG(MR)
C
        GO TO (100,200) ,K
 100    CONTINUE
C
CC      SOLVE R * Y = Y
C
        CALL SRXW(ADEC,Y,N,ND,ML,NLDIAG,MR,NRDIAG)
C
CC      MULTIPLY U = A * Y
C
        CALL SUAX(A,Y,U,N,ND,ML,NLDIAG,MR,NRDIAG)
 200    CONTINUE
C
CC
CC      SOLVE L * U = U
C
        CALL SLWU(ADEC,U,N,ND,ML,NLDIAG,MR,NRDIAG)
        CALL SCOPY(N,U,1,Y,1)
C
        RETURN
        END
LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN IDCG04
*DECK IDCG04
        SUBROUTINE SUAX(A,X,U,N,ND,ML,NLDIAG,MR,NRDIAG)
C
CC      MULTIPLIES  U = A * X
C
        DIMENSION A(N,ND),X(N),U(N),NLDIAG(ML),NRDIAG(MR)
C
CC      A         SPARSE MATRIX
CC      X         VECTOR TO BE MULTIPLIED WITH A
CC      U         RESULTING VECTOR
CC      N         NUMBER OF UNKNOWNS
CC      ND        =ML+MR+1
CC      ML        NUMBER OF NONZERO LOWER DIAGONALS
CC      NLDIAG    VECTOR POSITION OF LOWER DIAGONALS
CC      MR        NUMBER OF NONZERO UPPER DIAGONALS
CC      NRDIAG    VECTOR POSITION OF UPPER DIAGONALS
CC
CC      LOWER DIAGONALS
C
        CALL VZERO(U,N)
        DO  100  I=1,ML
        JSTART=NLDIAG(I)
        JNUM  =N-JSTART+1
        CALL SXYPZ(JNUM,A(JSTART,I),1,X(1),1,U(JSTART),1)
 100    CONTINUE
CC
CC      DIAGONAL
C
        CALL SXYPZ(N,A(1,ML+1),1,X(1),1,U(1),1)
C
CC      UPPER DIAGONALS
C
```

```
      DO  300  I=1,MR
      JSTART=NRDIAG(I)
      JNUM  =N-JSTART+1
      CALL SXYPZ(JNUM,A(1,I+ML+1),1,X(JSTART),1,U(1),1)
300   CONTINUE
C
      RETURN
      END
```

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN IDCG06

```
*DECK IDCG06
      SUBROUTINE SLWU(A,U,N,ND,ML,NLDIAG,MR,NRDIAG)
C
C     SOLVE  L * W = U
C
      DIMENSION A(N,ND),U(N),NLDIAG(ML),NRDIAG(MR)
C
C     A         MATRIX L IN LOWER DIAGONAL PART
C               MATRIX R IN UPPER DIAGONAL PART , DIAGONAL INCLUDED
C     U         RIGHT HAND SIDE AS INPUT
C               RESULTING VECTOR W AS OUTPUT
C     N         NUMBER OF UNKNOWNS
C     ND        =ML+MR+1
C     ML        NUMBER OF NONZERO LOWER DIAGONALS
C     NLDIAG    VECTOR POSITION OF LOWER DIAGONALS
C     MR        NUMBER OF NONZERO UPPER DIAGONALS
C     NRDIAG    VECTOR POSITION OF UPPER DIAGONALS
C
      DO  100  I=2,N
      U(I-1)=U(I-1)/A(I-1,ML+1)
      DO  100  J=1,ML
      JJ=I+NLDIAG(J)-2
      IF (JJ .GT. N) GO TO 100
      U(JJ)=U(JJ)-A(JJ,J)*U(I-1)
100   CONTINUE
      U(N)=U(N)/A(N,ML+1)
C
      RETURN
      END
```

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN IDCG07

```
*DECK IDCG07
      SUBROUTINE SRXW(A,U,N,ND,ML,NLDIAG,MR,NRDIAG)
C
C     SOLVE  R * X = W
C
      DIMENSION A(N,ND),U(N),NLDIAG(ML),NRDIAG(MR)
C
C     A         MATRIX L IN LOWER DIAGONAL PART
C               MATRIX R IN UPPER DIAGONAL PART , DIAGONAL INCLUDED
C     U         RIGHT HAND SIDE W AS INPUT
C               RESULTING VECTOR X AS OUTPUT
C     N         NUMBER OF UNKNOWNS
C     ND        =ML+MR+1
C     ML        NUMBER OF NONZERO LOWER DIAGONALS
C     NLDIAG    VECTOR POSITION OF LOWER DIAGONALS
C     MR        NUMBER OF NONZERO UPPER DIAGONALS
C     NRDIAG    VECTOR POSITION OF UPPER DIAGONALS
```

```
C
      DO  100  I=2,N
      II=N-I+2
      U(II)=U(II)/A(II,ML+1)
      DO  100  J=1,MR
      JJ=II-NRDIAG(J)+1
      IF (JJ .LT. 1) GO TO 100
      U(JJ)=U(JJ)-A(JJ,J+ML+1)*U(II)
 100  CONTINUE
      U(1)=U(1)/A(1,ML+1)
C
      RETURN
      END
```

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN MAT1

```
*DECK MAT1
      SUBROUTINE SAXPY(N, A, X, NX, Y, NY)                          SAX   10
      DIMENSION X(1), Y(1)                                          SAX   20
C                                                                   SAX   30
C THIS SUBROUTINE COMPUTES Y = Y + A*X.                             SAX   40
C                                                                   SAX   50
      IF (A.EQ.0.0 .OR. N.LE.0) RETURN                              SAX   60
      Y(1) = Y(1) + A*X(1)                                          SAX   70
      IF (N.EQ.1) RETURN                                            SAX   80
      NM1 = N - 1                                                   SAX   90
      DO 10 I=1,NM1                                                 SAX  100
         Y(I*NY+1) = Y(I*NY+1) + A*X(I*NX+1)                        SAX  110
   10 CONTINUE                                                      SAX  120
      RETURN                                                        SAX  130
      END                                                           SAX  140
```

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN MAT2

```
*DECK MAT2
      SUBROUTINE SCOPY(N, X, NX, Y, NY)                             SCO   10
      DIMENSION X(1), Y(1)                                          SCO   20
C                                                                   SCO   30
C THIS SUBROUTINE COPIES X ONTO Y.                                  SCO   40
C                                                                   SCO   50
      IF (N.LE.0) RETURN                                            SCO   60
      Y(1) = X(1)                                                   SCO   70
      IF (N.EQ.1) RETURN                                            SCO   80
      NM1 = N - 1                                                   SCO   90
      DO 10 I=1,NM1                                                 SCO  100
         Y(I*NY+1) = X(I*NX+1)                                      SCO  110
   10 CONTINUE                                                      SCO  120
      RETURN                                                        SCO  130
      END                                                           SCO  140
```

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN MAT3

```
*DECK MAT3
      FUNCTION SDOT(N, X, NX, Y, NY)                                SDO   10
      DIMENSION X(1), Y(1)                                          SDO   20
C THIS FUNCTION COMPUTES THE INNER PRODUCT OF X AND Y.              SDO   30
C                                                                   SDO   40
      SDOT = 0.0                                                    SDO   50
      IF (N.LE.0) RETURN                                            SDO   60
      SDOT = X(1)*Y(1)                                              SDO   70
```

```
      IF (N.EQ.1) RETURN                                          SDO    80
      NM1 = N - 1                                                 SDO    90
      DO 10 I=1,NM1                                               SDO   100
         SDOT = SDOT + X(I*NX+1)*Y(I*NY+1)                        SDO   110
   10 CONTINUE                                                    SDO   120
      RETURN                                                      SDO   130
      END                                                         SDO   140
```

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN MAT4

```
*DECK MAT4
      SUBROUTINE SSCAL(N, A, X, NX)                               SSC    10
      DIMENSION X(1)                                              SSC    20
C                                                                 SSC    30
C THIS SUBROUTINE SCALES THE VECTOR X BY A FACTOR A.              SSC    40
C                                                                 SSC    50
      IF (N.LE.0 .OR. A.EQ.1.0) RETURN                            SSC    60
      X(1) = A*X(1)                                               SSC    70
      IF (N.EQ.1) RETURN                                          SSC    80
      NM1 = N - 1                                                 SSC    90
      DO 10 I=1,NM1                                               SSC   100
         X(I*NX+1) = A*X(I*NX+1)                                  SSC   110
   10 CONTINUE                                                    SSC   120
      RETURN                                                      SSC   130
      END                                                         SSC   140
```

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN MAT5

```
*DECK MAT5
      SUBROUTINE SXYPZ(N, X, NX, Y, NY, Z, NZ)                    SXY    10
      DIMENSION X(1), Y(1), Z(1)                                  SXY    20
C                                                                 SXY    30
C THIS SUBROUTINE RETURNS Z = Z + X*Y  (ELEMENTWISE).             SXY    40
C                                                                 SXY    50
      IF (N.LE.0) RETURN                                          SXY    60
      Z(1) = Z(1) + X(1)*Y(1)                                     SXY    70
      IF (N.EQ.1) RETURN                                          SXY    80
      NM1 = N - 1                                                 SXY    90
      DO 10 I=1,NM1                                               SXY   100
         Z(I*NZ+1) = Z(I*NZ+1) + X(I*NX+1)*Y(I*NY+1)              SXY   110
   10 CONTINUE                                                    SXY   120
      RETURN                                                      SXY   130
      END                                                         SXY   140
```

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN MAT6

```
*DECK MAT6
      SUBROUTINE VZERO(X, N)                                      VZE    10
      DIMENSION X(1)                                              VZE    20
C                                                                 VZE    30
C THIS SUBROUTINE ZEROES A GIVEN VECTOR.                          VZE    40
C                                                                 VZE    50
      DO 10 I=1,N                                                 VZE    60
         X(I) = 0.0                                               VZE    70
   10 CONTINUE                                                    VZE    80
      RETURN                                                      VZE    90
      END                                                         VZE   100
```

```
*DECK OLYMP
      SUBROUTINE PAGE
      WRITE (6,1000)
 1000 FORMAT(1H1)
      RETURN
      END
C
      SUBROUTINE BLINES(K)
      DO 10 J=1,K
   10 WRITE (6,1000)
 1000 FORMAT(1H )
      RETURN
      END
C
      SUBROUTINE MESAGE(KMESS)
      DIMENSION KMESS(5)
      WRITE (6,1000) (KMESS(J),J=1,5)
 1000 FORMAT(5X,4A10,A8)
      RETURN
      END
C
      SUBROUTINE IVAR(KNAME,KVAL)
      WRITE (6,1000) KNAME,KVAL
 1000 FORMAT(/,1X,A8,3H = ,I10)
      RETURN
      END
C
      SUBROUTINE RVAR(KNAME,PVAL)
      WRITE (6,1000) KNAME,PVAL
 1000 FORMAT(/,1X,A8,3H = ,1PE13.4)
      RETURN
      END
C
      SUBROUTINE IARRAY(KNAME,KAR,KDIM)
      DIMENSION KAR(KDIM)
      WRITE (6,1000) KNAME
      WRITE (6,1001) (KAR(J),J=1,KDIM)
 1000 FORMAT(/,1X,A8)
 1001 FORMAT(1X,10I13)
      RETURN
      END
C
      SUBROUTINE RARRAY(KNAME,PAR,KDIM)
      DIMENSION PAR(KDIM)
      WRITE (6,1000) KNAME
      WRITE (6,1001) (PAR(J),J=1,KDIM)
 1000 FORMAT(/,1X,A8)
 1001 FORMAT(1X,1P10E13.4)
      RETURN
      END
C
      SUBROUTINE RESETI(KAR,KDIM,KVALUE)
      DIMENSION KAR(KDIM)
      DO 100 J=1,KDIM
  100 KAR(J)=KVALUE
      RETURN
      END
C
      SUBROUTINE RESETR(PAR,KDIM,PVALUE)
      DIMENSION PAR(KDIM)
```

```
      DO 100 J=1,KDIM
  100 PAR(J)=PVALUE
      RETURN
      END
C
      SUBROUTINE CPUTIM
      CALL SECOND(DUMMY)
      WRITE (6,1000) DUMMY
 1000 FORMAT (/,1X,22HCPU TIME USED SO FAR :,F10.4,/)
      RETURN
      END
C
      SUBROUTINE TITLE(ITITLE)
      DIMENSION ITITLE(4)
      DIMENSION L(10),IT(40)
C
  101 FORMAT (//,1X,4A10)
  102 FORMAT (/,1X,40A1,/)
  103 FORMAT (10A1)
C
      WRITE (6,101) ITITLE
      DO  100  J=1,40
  100 IT(J)=1H
      DO  120  J=1,4
      DECODE (10,103,ITITLE(5-J)) L(1),L(2),L(3),L(4),L(5),L(6),L(7),
     D                            L(8),L(9),L(10)
      JL=J
      DO  110  I=1,10
      IL=I
      IF (L(11-I) .NE. 1H ) GO TO 130
  110 CONTINUE
  120 CONTINUE
  130 CONTINUE
      KL=10*(5-JL)-IL+1
      DO  140  J=1,KL
  140 IT(J)=1H*
      WRITE (6,102) IT
C
      RETURN
      END
```

LOAD MAP - COURS CYBER LOADER 1.5-498 30/11/82 09.58.56. PAGE 1

FWA OF THE LOAD 111
LWA+1 OF THE LOAD 203310

TRANSFER ADDRESS -- COURS 6326

PROGRAM ENTRY POINTS -- COURS 6326

PROGRAM AND BLOCK ASSIGNMENTS.

BLOCK	ADDRESS	LENGTH	FILE	DATE	PROCSSR	VER	LEVEL	HARDWARE	COMMENTS
COURS	111	55315	LGO	30/11/82	FTN	4.8	498	6464 I	PROGRAM OPT=2
BAND	55426	56	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
BALDR	55504	221	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
BLDWU	55725	145	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
BRXW	56072	122	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
BUAX	56214	127	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
FATA	56343	133	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
SYMETR	56476	152	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
GS	56650	255	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
SLIDCG	57125	651	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
SAILR	57776	137	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
ATILDA	60135	137	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
SUAX	60274	131	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
SLWU	60425	53	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
SRXW	60500	60	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
SAXPY	60560	26	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
SCOPY	60606	22	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
SDOT	60630	27	LGO	30/11/82	FTN	4.8	498	6464 I	FUNCTION OPT=2
SSCAL	60657	24	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
SXYPZ	60703	27	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
VZERO	60732	12	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
PAGE	60744	12	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
BLINES	60756	25	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
MESAGE	61003	20	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
IVAR	61023	20	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
RVAR	61043	20	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
IARRAY	61063	33	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
RARRAY	61116	34	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
RESETI	61152	13	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
RESETR	61165	13	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
CPUTIM	61200	23	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
TITLE	61223	171	LGO	30/11/82	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
SGECO	61414	534	UL-LINPACK	16/04/81	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
SGESL	62150	216	UL-LINPACK	16/04/81	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
SGBFA	62366	264	UL-LINPACK	16/04/81	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
SASUM	62652	66	UL-LINPACK	16/04/81	FTN	4.8	498	6464 I	FUNCTION OPT=2
SGEFA	62740	171	UL-LINPACK	16/04/81	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
SGBSL	63131	233	UL-LINPACK	16/04/81	FTN	4.8	498	6464 I	SUBROUTINEOPT=2
ISAMAX	63364	42	UL-LINPACK	16/04/81	FTN	4.8	498	6464 I	FUNCTION OPT=2
TQL1	63426	253	UL-EISPACK	10/12/79	FTN	4.7	485	6464 I	SUBROUTINEOPT=2
TRED1	63701	227	UL-EISPACK	10/12/79	FTN	4.7	485	6464 I	SUBROUTINEOPT=2
FC1BMF	64130	305	UL-NAGF	12/05/80	FTN	4.7	485	6464 I	SUBROUTINEOPT=2 ROUND=+-*/
F04AVF	64435	162	UL-NAGF	12/05/80	FTN	4.7	485	6464 I	SUBROUTINEOPT=2 ROUND=+-*/
P01AAF	64617	160	UL-NAGF	23/06/80	FTN	4.7	485	6464 I	FUNCTION OPT=2 ROUND=+-*/
X02AAF	64777	12	UL-NAGF	23/06/80	FTN	4.7	485	6464 I	FUNCTION OPT=2 ROUND=+-*/
/AX04AA/	65011	1							
X04AAF	65012	15	UL-NAGF	23/06/80	FTN	4.7	485	6464 I	SUBROUTINEOPT=2 ROUND=+-*/
EXP.MSG	65027	16	SL-FORTRAN	30/10/80	COMPASS	3.6	518		COMMON ERROR MESSAGES FOR EXPONENTIATION.

LOAD MAP - COUPS CYBER LOADER 1.5-498 30/11/82 09.58.56. PAGE 2

BLOCK	ADDRESS	LENGTH	FILE	DATE	PROCSSR	VER	LEVEL	HARDWARE	COMMENTS
SQRT.	65045	32	SL-FORTRAN	30/10/80	COMPASS	3.6	518		COMPUTE THE SQUARE ROOT OF X. OPT=ALL.
SYS=1ST	65077	65	SL-FORTRAN	30/10/80	COMPASS	3.6	518		MATH LIBRARY LINK TO ERROR MESSAGE PROCESSOR.
FECMSK=	65164	41	SL-FORTRAN	31/10/80	COMPASS	3.6	518		INITIALIZE CONSTANTS.
/08.IO./	65225	135							
FLTIN=	65362	156	SL-FORTRAN	31/10/80	COMPASS	3.6	518		COMMON FLOATING INPUT CONVERTER.
/FCL.C./	65540	26							
FMTAP=	65566	357	SL-FORTRAN	31/10/80	COMPASS	3.6	518		CRACK APLIST AND FORMAT FOR KCDER/KRAKER.
FORUTL=	66145	75	SL-FORTRAN	31/10/80	COMPASS	3.6	518		FCL MISC. UTILITIES.
GETFIT=	66242	57	SL-FORTRAN	31/10/80	COMPASS	3.6	518		LOCATE AN FIT GIVEN A FILE NAME.
/IO.BUF./	66321	227							
INPB=	66550	356	SL-FORTRAN	31/10/80	COMPASS	3.6	518		BINARY READ FORTRAN RECORD.
KODER=	67126	465	SL-FORTRAN	31/10/80	COMPASS	3.6	518		OUTPUT FORMAT INTERPRETER.
OUTC=	67613	150	SL-FORTRAN	31/10/80	COMPASS	3.6	518		FORMATTED WRITE FORTRAN RECORD.
SPA=	67763	11	SL-FORTRAN	31/10/80	COMPASS	3.6	518		SPA= - SUBSTITUTE PARAMETER ADDRESSES.
SYSTEM	67774	26	SL-FORTRAN	31/10/80	COMPASS	3.6	518		USER CALLABLE ERROR PROCESSOR.
CLOCK=	70022	31	SL-FORTRAN	31/10/80	COMPASS	3.6	518		ACCESS SYSTEM CLOCKS FOR FORTRAN.
GOTOER=	70053	14	SL-FORTRAN	31/10/80	COMPASS	3.6	518		COMPUTED GO TO ERROR PROCESSOR.
REMARK	70067	4	SL-FORTRAN	31/10/80	COMPASS	3.6	518		ISSUE A MESSAGE TO THE DAYFILE.
SYSAID=	70073	1	SL-FORTRAN	30/10/80	COMPASS	3.6	518		LINK BETWEEN SYS=AID AND INITIALIZATION CODE.
SYS=AID	70074	7	SL-FORTRAN	30/10/80	COMPASS	3.6	518		AUXILIARY MATH LIBRARY LINK FOR ERRORS.
XTOI.	70103	63	SL-FORTRAN	30/10/80	COMPASS	3.6	518		REAL TO INTEGER EXPONENTIATION.
/STP.END/	70166	1							
02NTRY=	70167	1	SL-FORTRAN	31/10/80	COMPASS	3.6	518		FCL INITIALIZATION ROUTINE.
/FCL=ENT/	70170	40							
COMIO=	70230	33	SL-FORTRAN	31/10/80	COMPASS	3.6	518		COMMON CODED I/O ROUTINES AND CONSTANTS.
DECODE=	70263	73	SL-FORTRAN	31/10/80	COMPASS	3.6	518		FORMATTED READ FROM CORE.
FCL=FDL	70356	40	SL-FORTRAN	31/10/80	COMPASS	3.6	518		FCL CAPSULE LOADING
FEIFST=	70416	3	SL-FORTRAN	31/10/80	COMPASS	3.6	518		CONVERTED DATA STORAGE
FLTOUT=	70421	311	SL-FORTRAN	31/10/80	COMPASS	3.6	518		COMMON FLOATING OUTPUT CODE
FORSYS=	70732	455	SL-FORTRAN	31/10/80	COMPASS	3.6	518		FORTRAN OBJECT LIBRARY UTILITIES.
INCOM=	71407	145	SL-FORTRAN	31/10/80	COMPASS	3.6	518		COMMON INPUT FORMATTING CODE
KRAKER=	71554	375	SL-FORTRAN	31/10/80	COMPASS	3.6	518		PROCESS FORMATTED FORTRAN INPUT.
OUTB=	72151	174	SL-FORTRAN	31/10/80	COMPASS	3.6	518		BINARY WRITE FORTRAN RECORD.
OUTCOM=	72345	154	SL-FORTRAN	31/10/80	COMPASS	3.6	518		COMMON OUTPUT CODE
REWIND=	72521	33	SL-FORTRAN	31/10/80	COMPASS	3.6	518		POSITION FILE AT BEGINNING-OF-INFORMATION.
CTL$PTL	72554	34	SL-SYSLIB	06/03/80	COMPASS	3.6	485		CRM CONTROLLER - PARTIAL GET/PUT.
CTL$SKP	72610	56	SL-SYSLIB	06/03/80	COMPASS	3.6	485		CRM CONTROLLER - SKIP PHYSICAL/FILE.
CTL$RM	72666	376	SL-SYSLIB	06/03/80	COMPASS	3.6	485		CRM CONTROLLING ROUTINE.
CTL$WR	73264	40	SL-SYSLIB	06/03/80	COMPASS	3.6	485		CRM CONTROLLER - WEOX, REWIND
ERR$RM	73324	25	SL-SYSLIB	06/03/80	COMPASS	3.6	485		CRM ERROR PROCESSOR ENTRY.
LIST$RM	73351	67	SL-SYSLIB	06/03/80	COMPASS	3.6	485		CRM - ALLOCATE SPACE FOR LIST OF FILES
/FDL.COM/	73440	14							
FDL.RES	73454	211	SL-SYSLIB	20/08/80	COMPASS	3.6	498		FAST DYNAMIC LOADER RESIDENT.
FDL.MMI	73665	222	SL-SYSLIB	20/08/80	COMPASS	3.6	498		FDL MEMORY MANAGER INTERFACE.
CMF.ALF	74107	162	SL-SYSLIB	03/11/80	COMPASS	3.6	518		CMM V1.1 - ALLOCATE FIXED.
CMF.CSF	74271	6	SL-SYSLIB	03/11/80	COMPASS	3.6	518		CMM V1.1 - CHANGE SPECS FIXED.
CMM.FFA	74277	14	SL-SYSLIB	03/11/80	COMPASS	3.6	518		CMM V1.1 - FIXED FREE ALGORITHM.
CMF.FRF	74313	36	SL-SYSLIB	03/11/80	COMPASS	3.6	518		CMM V1.1 - FREE FIXED.
CMF.GSS	74351	22	SL-SYSLIB	03/11/80	COMPASS	3.6	518		CMM V1.1 - GET SUMMARY STATISTICS.
CMM.KIL	74373	12	SL-SYSLIB	03/11/80	COMPASS	3.6	518		CMM V1.1 - DEACTIVATE CMM.
CMM.MFM	74405	7	SL-SYSLIB	03/11/80	COMPASS	3.6	518		
CMM.R	74414	206	SL-SYSLIB	03/11/80	COMPASS	3.6	518		CMM V1.1 - RESIDENT SUBROUTINES.
CMF.SLF	74622	22	SL-SYSLIB	03/11/80	COMPASS	3.6	518		CMM V1.1 - SHRINK AT LWA FIXED.
CPU.SYS	74644	40	SL-SYSLIB	30/10/80	COMPASS	3.6	498		PROCESS SYSTEM REQUEST.
//	74704	106404							

2.580 CP SECONDS 113000B CM STORAGE USED 87 TABLE MOVES

SOLUTION ANALYTIQUE DE SOLOVEV
****

XANALYT =

-1.3673E-02	-1.3195E-02	-1.1759E-02	-9.3667E-03	-6.0173E-03	-1.7110E-03	0.	0.	0.	0.
-2.7222E-02	-2.6686E-02	-2.5078E-02	-2.2399E-02	-1.8649E-02	-1.3826E-02	-7.9320E-03	-9.6622E-04	0.	0.
-4.0436E-02	-3.9839E-02	-3.8050E-02	-3.5068E-02	-3.0893E-02	-2.5525E-02	-1.8965E-02	-1.1211E-02	-2.2649E-03	0.
-5.3133E-02	-5.2473E-02	-5.0492E-02	-4.7191E-02	-4.2569E-02	-3.6627E-02	-2.9364E-02	-2.0780E-02	-1.0877E-02	0.
-6.5122E-02	-6.4395E-02	-6.2213E-02	-5.8576E-02	-5.3465E-02	-4.6939E-02	-3.8938E-02	-2.9483E-02	-1.8573E-02	-6.2077E-03
-7.6202E-02	-7.5404E-02	-7.3011E-02	-6.9023E-02	-6.3439E-02	-5.6260E-02	-4.7486E-02	-3.7116E-02	-2.5151E-02	-1.1591E-02
-8.6161E-02	-8.5290E-02	-8.2676E-02	-7.8320E-02	-7.2221E-02	-6.4360E-02	-5.4796E-02	-4.3470E-02	-3.0402E-02	-1.5591E-02
-9.4779E-02	-9.3831E-02	-9.0986E-02	-8.6246E-02	-7.9610E-02	-7.1077E-02	-6.0649E-02	-4.8324E-02	-3.4103E-02	-1.7986E-02
-1.0182E-01	-1.0080E-01	-9.7712E-02	-9.2572E-02	-8.5375E-02	-7.6122E-02	-6.4812E-02	-5.1447E-02	-3.6025E-02	-1.8547E-02
-1.0706E-01	-1.0595E-01	-1.0261E-01	-9.7056E-02	-8.9276E-02	-7.9273E-02	-6.7047E-02	-5.2599E-02	-3.5927E-02	-1.7033E-02
-1.1023E-01	-1.0903E-01	-1.0544E-01	-9.9448E-02	-9.1063E-02	-8.0281E-02	-6.7103E-02	-5.1529E-02	-3.3560E-02	-1.3194E-02
-1.1108E-01	-1.0979E-01	-1.0593E-01	-9.9490E-02	-9.0476E-02	-7.8886E-02	-6.4720E-02	-4.7979E-02	-2.6663E-02	-6.7707E-03
-1.0934E-01	-1.0796E-01	-1.0382E-01	-9.6911E-02	-8.7246E-02	-7.4818E-02	-5.9630E-02	-4.1679E-02	-2.0967E-02	0.
-1.0473E-01	-1.0325E-01	-9.8819E-02	-9.1433E-02	-8.1093E-02	-6.7799E-02	-5.1551E-02	-3.2349E-02	-1.0193E-02	0.
-9.6956E-02	-9.5379E-02	-9.0649E-02	-8.2766E-02	-7.1730E-02	-5.7540E-02	-4.0197E-02	-1.9701E-02	0.	0.
-8.5726E-02	-8.4047E-02	-7.9009E-02	-7.0612E-02	-5.8856E-02	-4.3741E-02	-2.5260E-02	-3.4356E-03	0.	0.
-7.0731E-02	-6.8945E-02	-6.3589E-02	-5.4662E-02	-4.2164E-02	-2.6095E-02	-6.4553E-03	0.	0.	0.
-5.1650E-02	-4.9755E-02	-4.4071E-02	-3.4598E-02	-2.1335E-02	-4.2828E-03	0.	0.	0.	0.
-2.8157E-02	-2.6150E-02	-2.0128E-02	-1.0092E-02	0.	0.	0.	0.	0.	0.

DECOMPOSE A=LOR AVEC EALOR DE MATLIB
****

CPU TIME USED SO FAR : 21.6210

CPU TIME USED SO FAR : 21.9030

RESOUD LOR*X=B AVEC BAND DE MATLIB
****

CPU TIME USED SO FAR : 21.9680

X =

-1.3677E-02	-1.3197E-02	-1.1758E-02	-9.3621E-03	-6.0109E-03	-1.7079E-03	0.	0.	0.	0.
-2.7230E-02	-2.6694E-02	-2.5083E-02	-2.2401E-02	-1.8649E-02	-1.3827E-02	-7.9272E-03	-9.6414E-04	0.	0.
-4.0449E-02	-3.9851E-02	-3.8060E-02	-3.5076E-02	-3.0899E-02	-2.5530E-02	-1.8966E-02	-1.1212E-02	-2.2604E-03	0.
-5.3150E-02	-5.2489E-02	-5.0507E-02	-4.7204E-02	-4.2560E-02	-3.6636E-02	-2.9370E-02	-2.0765E-02	-1.0877E-02	0.
-6.5143E-02	-6.4415E-02	-6.2232E-02	-5.8593E-02	-5.3500E-02	-4.6952E-02	-3.8948E-02	-2.9490E-02	-1.8577E-02	-6.2083E-03
-7.6226E-02	-7.5428E-02	-7.3034E-02	-6.9044E-02	-6.3458E-02	-5.6277E-02	-4.7500E-02	-3.7127E-02	-2.5158E-02	-1.1594E-02
-8.6188E-02	-8.5317E-02	-8.2702E-02	-7.8344E-02	-7.2243E-02	-6.4400E-02	-5.4813E-02	-4.3494E-02	-3.0411E-02	-1.5596E-02
-9.4809E-02	-9.3860E-02	-9.1015E-02	-8.6273E-02	-7.9635E-02	-7.1099E-02	-6.0668E-02	-4.8339E-02	-3.4114E-02	-1.7992E-02
-1.0186E-01	-1.0083E-01	-9.7743E-02	-9.2600E-02	-8.5401E-02	-7.6146E-02	-6.4833E-02	-5.1464E-02	-3.6037E-02	-1.8554E-02
-1.0709E-01	-1.0596E-01	-1.0264E-01	-9.7086E-02	-8.9304E-02	-7.9298E-02	-6.7069E-02	-5.2616E-02	-3.5940E-02	-1.7040E-02
-1.1026E-01	-1.0907E-01	-1.0547E-01	-9.9480E-02	-9.1091E-02	-8.0307E-02	-6.7125E-02	-5.1547E-02	-3.3573E-02	-1.3201E-02
-1.1111E-01	-1.0983E-01	-1.0596E-01	-9.9522E-02	-9.0505E-02	-7.8912E-02	-6.4742E-02	-4.7997E-02	-2.8675E-02	-6.7788E-03
-1.0937E-01	-1.0799E-01	-1.0385E-01	-9.6943E-02	-8.7274E-02	-7.4644E-02	-5.9651E-02	-4.1695E-02	-2.0978E-02	0.
-1.0476E-01	-1.0328E-01	-9.8850E-02	-9.1463E-02	-8.1121E-02	-6.7823E-02	-5.1571E-02	-3.2363E-02	-1.0204E-02	0.
-9.6986E-02	-9.5410E-02	-9.0679E-02	-8.2795E-02	-7.1755E-02	-5.7562E-02	-4.0214E-02	-1.9712E-02	0.	0.
-8.5754E-02	-8.4075E-02	-7.9037E-02	-7.0638E-02	-5.8879E-02	-4.3760E-02	-2.5263E-02	-3.4433E-03	0.	0.
-7.0754E-02	-6.8970E-02	-6.3615E-02	-5.4686E-02	-4.2134E-02	-2.6110E-02	-6.4676E-03	0.	0.	0.
-5.1668E-02	-4.9775E-02	-4.4094E-02	-3.4619E-02	-2.1351E-02	-4.2919E-03	0.	0.	0.	0.
-2.8168E-02	-2.6164E-02	-2.0149E-02	-1.0110E-02	0.	0.	0.	0.	0.	0.

DECOMPOSE A=LDR AVEC SGBFA DE LINPACK
****

CPU TIME USED SO FAR : 22.0420

CPU TIME USED SO FAR : 22.3900

RESOUD LDR*X=B AVEC SGBSL DE LINPACK
****

IPIVOT =

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150
151	152	153	154	155	156	157	158	159	160
161	162	163	164	165	166	167	168	169	170
171	172	173	174	175	176	177	178	179	180
181	182	183	184	185	186	187	188	189	190

CPU TIME USED SO FAR : 22.4410

CPU TIME USED SO FAR : 22.5230

X =

-1.3677E-02	-1.3197E-02	-1.1758E-02	-9.3621E-03	-6.0109E-03	-1.7079E-03	0.	0.	0.	0.
-2.7230E-02	-2.6694E-02	-2.5083E-02	-2.2401E-02	-1.8649E-02	-1.3827E-02	-7.9272E-03	-9.6414E-04	0.	0.
-4.0449E-02	-3.9851E-02	-3.8060E-02	-3.5076E-02	-3.0899E-02	-2.5530E-02	-1.8966E-02	-1.1212E-02	-2.2604E-03	0.
-5.3150E-02	-5.2489E-02	-5.0507E-02	-4.7204E-02	-4.2580E-02	-3.6636E-02	-2.9370E-02	-2.0785E-02	-1.0877E-02	0.
-6.5143E-02	-6.4415E-02	-6.2232E-02	-5.8593E-02	-5.3500E-02	-4.6952E-02	-3.8946E-02	-2.9490E-02	-1.8577E-02	-6.2083E-03
-7.6226E-02	-7.5428E-02	-7.3034E-02	-6.9044E-02	-6.3458E-02	-5.6277E-02	-4.7500E-02	-3.7127E-02	-2.5158E-02	-1.1594E-02
-8.6188E-02	-8.5317E-02	-8.2702E-02	-7.8344E-02	-7.2243E-02	-6.4400E-02	-5.4813E-02	-4.3484E-02	-3.0411E-02	-1.5596E-02
-9.4809E-02	-9.3860E-02	-9.1015E-02	-8.6273E-02	-7.9635E-02	-7.1099E-02	-6.0668E-02	-4.8339E-02	-3.4114E-02	-1.7992E-02
-1.0186E-01	-1.0083E-01	-9.7743E-02	-9.2600E-02	-8.5401E-02	-7.6146E-02	-6.4833E-02	-5.1464E-02	-3.6037E-02	-1.8554E-02
-1.0709E-01	-1.0598E-01	-1.0264E-01	-9.7086E-02	-8.9304E-02	-7.9298E-02	-6.7069E-02	-5.2616E-02	-3.5940E-02	-1.7040E-02
-1.1026E-01	-1.0907E-01	-1.0547E-01	-9.9480E-02	-9.1091E-02	-8.0307E-02	-6.7125E-02	-5.1547E-02	-3.3573E-02	-1.3201E-02
-1.1111E-01	-1.0983E-01	-1.0596E-01	-9.9522E-02	-9.0505E-02	-7.8912E-02	-6.4742E-02	-4.7997E-02	-2.8675E-02	-6.7788E-03
-1.0937E-01	-1.0799E-01	-1.0385E-01	-9.6943E-02	-8.7274E-02	-7.4844E-02	-5.9651E-02	-4.1695E-02	-2.0978E-02	0.
-1.0476E-01	-1.0328E-01	-9.8850E-02	-9.1463E-02	-8.1121E-02	-6.7823E-02	-5.1571E-02	-3.2363E-02	-1.0204E-02	0.
-9.6986E-02	-9.5410E-02	-9.3679E-02	-8.2795E-02	-7.1755E-02	-5.7562E-02	-4.0214E-02	-1.9712E-02	0.	0.
-8.5754E-02	-8.4075E-02	-7.9037E-02	-7.0638E-02	-5.8679E-02	-4.3760E-02	-2.5283E-02	-3.4433E-03	0.	0.
-7.0754E-02	-6.8970E-02	-6.3615E-02	-5.4686E-02	-4.2184E-02	-2.6110E-02	-6.4678E-03	0.	0.	0.
-5.1668E-02	-4.9775E-02	-4.4094E-02	-3.4619E-02	-2.1351E-02	-4.2919E-03	0.	0.	0.	0.
-2.8168E-02	-2.6164E-02	-2.0149E-02	-1.0110E-02	0.	0.	0.	0.	0.	0.

DECOMPOSE MATRICE PLEINE AVEC SGECO
****

CPU TIME USED SO FAR : 23.0600

CPU TIME USED SO FAR : 26.6500

COND(A) = = 5.3121E-05

RESOUD MATRICE PLEINE AVEC SGESL
****

CPU TIME USED SO FAR : 26.6600

CPU TIME USED SO FAR : 26.9860

X =

-1.3677E-02	-1.3197E-02	-1.1758E-02	-9.3621E-03	-6.0109E-03	-1.7079E-03	0.	0.	0.	0.
-2.7230E-02	-2.6694E-02	-2.5083E-02	-2.2401E-02	-1.8649E-02	-1.3827E-02	-7.9272E-03	-9.6414E-04	0.	0.
-4.0449E-02	-3.9851E-02	-3.8060E-02	-3.5076E-02	-3.0899E-02	-2.5530E-02	-1.8966E-02	-1.1212E-02	-2.2604E-03	0.
-5.3150E-02	-5.2489E-02	-5.0507E-02	-4.7204E-02	-4.2560E-02	-3.6636E-02	-2.9370E-02	-2.0785E-02	-1.0877E-02	0.
-6.5143E-02	-6.4415E-02	-6.2232E-02	-5.8593E-02	-5.3500E-02	-4.6952E-02	-3.8948E-02	-2.9490E-02	-1.8577E-02	-6.2083E-03
-7.6226E-02	-7.5428E-02	-7.3034E-02	-6.9044E-02	-6.3458E-02	-5.6277E-02	-4.7500E-02	-3.7127E-02	-2.5158E-02	-1.1594E-02
-8.6188E-02	-8.5317E-02	-8.2702E-02	-7.8344E-02	-7.2243E-02	-6.4400E-02	-5.4813E-02	-4.3484E-02	-3.0411E-02	-1.5596E-02
-9.4809E-02	-9.3860E-02	-9.1015E-02	-8.6273E-02	-7.9635E-02	-7.1099E-02	-6.0668E-02	-4.8339E-02	-3.4114E-02	-1.7992E-02
-1.0166E-01	-1.0083E-01	-9.7743E-02	-9.2600E-02	-8.5401E-02	-7.6146E-02	-6.4833E-02	-5.1464E-02	-3.6037E-02	-1.8554E-02
-1.0709E-01	-1.0598E-01	-1.0264E-01	-9.7086E-02	-8.9304E-02	-7.9298E-02	-6.7069E-02	-5.2616E-02	-3.5940E-02	-1.7040E-02
-1.1026E-01	-1.0907E-01	-1.0547E-01	-9.9480E-02	-9.1091E-02	-8.0307E-02	-6.7125E-02	-5.1547E-02	-3.3573E-02	-1.3201E-02
-1.1111E-01	-1.0983E-01	-1.0596E-01	-9.9522E-02	-9.0505E-02	-7.8912E-02	-6.4742E-02	-4.7997E-02	-2.8675E-02	-6.7788E-03
-1.0937E-01	-1.0799E-01	-1.0385E-01	-9.6943E-02	-8.7274E-02	-7.4644E-02	-5.9651E-02	-4.1695E-02	-2.0978E-02	0.
-1.0476E-01	-1.0328E-01	-9.8850E-02	-9.1463E-02	-8.1121E-02	-6.7823E-02	-5.1571E-02	-3.2363E-02	-1.0204E-02	0.
-9.6986E-02	-9.5410E-02	-9.0679E-02	-8.2795E-02	-7.1755E-02	-5.7562E-02	-4.0214E-02	-1.9712E-02	0.	0.
-8.5754E-02	-8.4075E-02	-7.9037E-02	-7.0638E-02	-5.8879E-02	-4.3760E-02	-2.5283E-02	-3.4433E-03	0.	0.
-7.0754E-02	-6.8970E-02	-6.3615E-02	-5.4686E-02	-4.2184E-02	-2.6110E-02	-6.4678E-03	0.	0.	0.
-5.1668E-02	-4.9775E-02	-4.4094E-02	-3.4619E-02	-2.1351E-02	-4.2919E-03	0.	0.	0.	0.
-2.8168E-02	-2.6164E-02	-2.0149E-02	-1.0110E-02	0.	0.	0.	0.	0.	0.

DECOMPOSE A=LDR AVEC F01BMF DE NAG
****

CPU TIME USED SO FAR : 27.0660

CPU TIME USED SO FAR : 27.6150

RESOUD LDR*X=B AVEC F04AVF DE NAG
****

CPU TIME USED SO FAR : 27.6220

CPU TIME USED SO FAR : 27.7060

X =

-1.3677E-02	-1.3197E-02	-1.1758E-02	-9.3621E-03	-6.0109E-03	-1.7079E-03	0.	0.	0.	0.
-2.7230E-02	-2.6694E-02	-2.5083E-02	-2.2401E-02	-1.8649E-02	-1.3827E-02	-7.9272E-03	-9.6414E-04	0.	0.
-4.0449E-02	-3.9851E-02	-3.8060E-02	-3.5076E-02	-3.0899E-02	-2.5530E-02	-1.8966E-02	-1.1212E-02	-2.2604E-03	0.
-5.3150E-02	-5.2489E-02	-5.0507E-02	-4.7204E-02	-4.2560E-02	-3.6636E-02	-2.9370E-02	-2.0785E-02	-1.0877E-02	0.
-6.5143E-02	-6.4415E-02	-6.2232E-02	-5.8593E-02	-5.3500E-02	-4.6952E-02	-3.8948E-02	-2.9490E-02	-1.8577E-02	-6.2083E-03
-7.6226E-02	-7.5428E-02	-7.3034E-02	-6.9044E-02	-6.3458E-02	-5.6277E-02	-4.7500E-02	-3.7127E-02	-2.5158E-02	-1.1594E-02
-8.6188E-02	-8.5317E-02	-8.2702E-02	-7.8344E-02	-7.2243E-02	-6.4400E-02	-5.4813E-02	-4.3484E-02	-3.0411E-02	-1.5596E-02
-9.4809E-02	-9.3860E-02	-9.1015E-02	-8.6273E-02	-7.9635E-02	-7.1099E-02	-6.0668E-02	-4.8339E-02	-3.4114E-02	-1.7992E-02
-1.0166E-01	-1.0083E-01	-9.7743E-02	-9.2600E-02	-8.5401E-02	-7.6146E-02	-6.4833E-02	-5.1464E-02	-3.6037E-02	-1.8554E-02
-1.0709E-01	-1.0598E-01	-1.0264E-01	-9.7086E-02	-8.9304E-02	-7.9298E-02	-6.7069E-02	-5.2616E-02	-3.5940E-02	-1.7040E-02
-1.1026E-01	-1.0907E-01	-1.0547E-01	-9.9480E-02	-9.1091E-02	-8.0307E-02	-6.7125E-02	-5.1547E-02	-3.3573E-02	-1.3201E-02
-1.1111E-01	-1.0983E-01	-1.0596E-01	-9.9522E-02	-9.0505E-02	-7.8912E-02	-6.4742E-02	-4.7997E-02	-2.8675E-02	-6.7788E-03
-1.0937E-01	-1.0799E-01	-1.0385E-01	-9.6943E-02	-8.7274E-02	-7.4644E-02	-5.9651E-02	-4.1695E-02	-2.0978E-02	0.
-1.0476E-01	-1.0328E-01	-9.8850E-02	-9.1463E-02	-8.1121E-02	-6.7823E-02	-5.1571E-02	-3.2363E-02	-1.0204E-02	0.
-9.6986E-02	-9.5410E-02	-9.0679E-02	-8.2795E-02	-7.1755E-02	-5.7562E-02	-4.0214E-02	-1.9712E-02	0.	0.
-8.5754E-02	-8.4075E-02	-7.9037E-02	-7.0638E-02	-5.8879E-02	-4.3760E-02	-2.5283E-02	-3.4433E-03	0.	0.
-7.0754E-02	-6.8970E-02	-6.3615E-02	-5.4686E-02	-4.2184E-02	-2.6110E-02	-6.4678E-03	0.	0.	0.
-5.1668E-02	-4.9775E-02	-4.4094E-02	-3.4619E-02	-2.1351E-02	-4.2919E-03	0.	0.	0.	0.
-2.8168E-02	-2.6164E-02	-2.0149E-02	-1.0110E-02	0.	0.	0.	0.	0.	0.

GAUSS - SEIDEL
****

CPU TIME USED SO FAR : 27.7900

CPU TIME USED SO FAR : 34.0940

ITER = = 348

X =

-1.3677E-02	-1.3197E-02	-1.1758E-02	-9.3618E-03	-6.0107E-03	-1.7078E-03	0.	0.	0.	0.	
-2.7229E-02	-2.6692E-02	-2.5082E-02	-2.2400E-02	-1.8648E-02	-1.3826E-02	-7.9270E-03	-9.6412E-04	0.	0.	
-4.0447E-02	-3.9850E-02	-3.8059E-02	-3.5074E-02	-3.0898E-02	-2.5529E-02	-1.8966E-02	-1.1212E-02	-2.2604E-03	0.	
-5.3148E-02	-5.2487E-02	-5.0505E-02	-4.7202E-02	-4.2578E-02	-3.6635E-02	-2.9369E-02	-2.0784E-02	-1.0877E-02	0.	
-6.5140E-02	-6.4412E-02	-6.2229E-02	-5.8591E-02	-5.3498E-02	-4.6950E-02	-3.8947E-02	-2.9490E-02	-1.8576E-02	-6.2082E-03	
-7.6223E-02	-7.5425E-02	-7.3031E-02	-6.9041E-02	-6.3456E-02	-5.6275E-02	-4.7498E-02	-3.7126E-02	-2.5158E-02	-1.1594E-02	
-8.6185E-02	-8.5313E-02	-8.2699E-02	-7.8341E-02	-7.2241E-02	-6.4398E-02	-5.4812E-02	-4.3463E-02	-3.0411E-02	-1.5595E-02	
-9.4805E-02	-9.3656E-02	-9.1011E-02	-8.6270E-02	-7.9632E-02	-7.1097E-02	-6.0666E-02	-4.8338E-02	-3.4114E-02	-1.7992E-02	
-1.0185E-01	-1.0082E-01	-9.7739E-02	-9.2597E-02	-8.5399E-02	-7.6143E-02	-6.4831E-02	-5.1463E-02	-3.6037E-02	-1.8554E-02	
-1.0709E-01	-1.0598E-01	-1.0264E-01	-9.7083E-02	-8.9301E-02	-7.9296E-02	-6.7067E-02	-5.2615E-02	-3.5940E-02	-1.7040E-02	
-1.1026E-01	-1.0906E-01	-1.0547E-01	-9.9476E-02	-9.1089E-02	-8.0304E-02	-6.7124E-02	-5.1546E-02	-3.3572E-02	-1.3201E-02	
-1.1111E-01	-1.0982E-01	-1.0596E-01	-9.9518E-02	-9.0502E-02	-7.8910E-02	-6.4741E-02	-4.7996E-02	-2.8675E-02	-6.7787E-03	
-1.0937E-01	-1.0799E-01	-1.0384E-01	-9.6940E-02	-8.7272E-02	-7.4642E-02	-5.9649E-02	-4.1694E-02	-2.0978E-02	0.	
-1.0476E-01	-1.0328E-01	-9.8847E-02	-9.1461E-02	-8.1119E-02	-6.7822E-02	-5.1570E-02	-3.2363E-02	-1.0204E-02	0.	
-9.6983E-02	-9.5407E-02	-9.0677E-02	-8.2792E-02	-7.1753E-02	-5.7560E-02	-4.0214E-02	-1.9711E-02	0.	0.	
-8.5752E-02	-8.4073E-02	-7.9035E-02	-7.0636E-02	-5.8878E-02	-4.3759E-02	-2.5282E-02	-3.4433E-03	0.	0.	
-7.0752E-02	-6.8968E-02	-6.3613E-02	-5.4684E-02	-4.2183E-02	-2.6109E-02	-6.4677E-03	0.	0.	0.	
-5.1667E-02	-4.9774E-02	-4.4093E-02	-3.4618E-02	-2.1351E-02	-4.2918E-03	0.	0.	0.	0.	
-2.8167E-02	-2.6164E-02	-2.0149E-02	-1.0110E-02	0.	0.	0.	0.	0.	0.	

GAUSS - SEIDEL AVEC SURRELAXATION
****

CPU TIME USED SO FAR : 34.1790

CPU TIME USED SO FAR : 34.9610

OMEGA = = 1.7700E+00

ITER = = 43

X =

-1.3678E-02	-1.3196E-02	-1.1759E-02	-9.3617E-03	-6.0101E-03	-1.7078E-03	0.	0.	0.	0.	
-2.7230E-02	-2.6693E-02	-2.5082E-02	-2.2399E-02	-1.8647E-02	-1.3826E-02	-7.9273E-03	-9.6417E-04	0.	0.	
-4.0449E-02	-3.9851E-02	-3.8059E-02	-3.5074E-02	-3.0898E-02	-2.5530E-02	-1.8966E-02	-1.1212E-02	-2.2604E-03	0.	
-5.3150E-02	-5.2489E-02	-5.0506E-02	-4.7202E-02	-4.2579E-02	-3.6636E-02	-2.9370E-02	-2.0785E-02	-1.0877E-02	0.	
-6.5143E-02	-6.4415E-02	-6.2231E-02	-5.8592E-02	-5.3500E-02	-4.6952E-02	-3.8949E-02	-2.9491E-02	-1.8577E-02	-6.2084E-03	
-7.6227E-02	-7.5426E-02	-7.3034E-02	-6.9043E-02	-6.3458E-02	-5.6277E-02	-4.7500E-02	-3.7127E-02	-2.5159E-02	-1.1594E-02	
-8.6189E-02	-8.5317E-02	-8.2702E-02	-7.8344E-02	-7.2244E-02	-6.4400E-02	-5.4814E-02	-4.3484E-02	-3.0411E-02	-1.5596E-02	
-9.4810E-02	-9.3861E-02	-9.1016E-02	-8.6274E-02	-7.9635E-02	-7.1100E-02	-6.0668E-02	-4.8340E-02	-3.4115E-02	-1.7992E-02	
-1.0186E-01	-1.0083E-01	-9.7744E-02	-9.2601E-02	-8.5402E-02	-7.6147E-02	-6.4834E-02	-5.1464E-02	-3.6038E-02	-1.8554E-02	
-1.0709E-01	-1.0598E-01	-1.0265E-01	-9.7087E-02	-8.9305E-02	-7.9299E-02	-6.7070E-02	-5.2617E-02	-3.5941E-02	-1.7040E-02	
-1.1027E-01	-1.0907E-01	-1.0547E-01	-9.9481E-02	-9.1093E-02	-8.0308E-02	-6.7126E-02	-5.1543E-02	-3.3573E-02	-1.3201E-02	
-1.1112E-01	-1.0983E-01	-1.0596E-01	-9.9523E-02	-9.0506E-02	-7.8912E-02	-6.4743E-02	-4.7997E-02	-2.8676E-02	-6.7789E-03	
-1.0937E-01	-1.0799E-01	-1.0385E-01	-9.6944E-02	-8.7275E-02	-7.4844E-02	-5.9651E-02	-4.1696E-02	-2.0978E-02	0.	
-1.0476E-01	-1.0328E-01	-9.8852E-02	-9.1464E-02	-8.1121E-02	-6.7824E-02	-5.1571E-02	-3.2364E-02	-1.0204E-02	0.	
-9.6988E-02	-9.5411E-02	-9.0681E-02	-8.2795E-02	-7.1756E-02	-5.7562E-02	-4.0215E-02	-1.9712E-02	0.	0.	
-8.5755E-02	-8.4076E-02	-7.9038E-02	-7.0639E-02	-5.8879E-02	-4.3760E-02	-2.5283E-02	-3.4433E-03	0.	0.	
-7.0755E-02	-6.8970E-02	-6.3615E-02	-5.4686E-02	-4.2184E-02	-2.6110E-02	-6.4678E-03	0.	0.	0.	
-5.1669E-02	-4.9776E-02	-4.4095E-02	-3.4619E-02	-2.1351E-02	-4.2919E-03	0.	0.	0.	0.	
-2.8168E-02	-2.6164E-02	-2.0149E-02	-1.0110E-02	0.	0.	0.	0.	0.	0.	

GRADIENTS CONJUGUES
****

CPU TIME USED SO FAR : 35.0420

CPU TIME USED SO FAR : 40.3960

ITER = = 163

X =

-1.3677E-02	-1.3197E-02	-1.1758E-02	-9.3621E-03	-6.0109E-03	-1.7079E-03	0.	0.	0.	0.
-2.7230E-02	-2.6694E-02	-2.5083E-02	-2.2401E-02	-1.8649E-02	-1.3827E-02	-7.9272E-03	-9.6414E-04	0.	0.
-4.0449E-02	-3.9851E-02	-3.8060E-02	-3.5076E-02	-3.0899E-02	-2.5530E-02	-1.8966E-02	-1.1212E-02	-2.2604E-03	0.
-5.3150E-02	-5.2489E-02	-5.0507E-02	-4.7204E-02	-4.2580E-02	-3.6636E-02	-2.9370E-02	-2.0785E-02	-1.0877E-02	0.
-6.5143E-02	-6.4415E-02	-6.2232E-02	-5.8593E-02	-5.3500E-02	-4.6952E-02	-3.8948E-02	-2.9490E-02	-1.8577E-02	-6.2083E-03
-7.6226E-02	-7.5428E-02	-7.3034E-02	-6.9044E-02	-6.3458E-02	-5.6277E-02	-4.7500E-02	-3.7127E-02	-2.5158E-02	-1.1594E-02
-8.6188E-02	-8.5317E-02	-8.2702E-02	-7.8344E-02	-7.2243E-02	-6.4400E-02	-5.4813E-02	-4.3484E-02	-3.0411E-02	-1.5596E-02
-9.4809E-02	-9.3860E-02	-9.1015E-02	-8.6273E-02	-7.9635E-02	-7.1099E-02	-6.0668E-02	-4.8339E-02	-3.4114E-02	-1.7992E-02
-1.0186E-01	-1.0083E-01	-9.7743E-02	-9.2600E-02	-8.5401E-02	-7.6146E-02	-6.4833E-02	-5.1464E-02	-3.6037E-02	-1.8554E-02
-1.0709E-01	-1.0598E-01	-1.0264E-01	-9.7086E-02	-8.9304E-02	-7.9298E-02	-6.7069E-02	-5.2616E-02	-3.5940E-02	-1.7040E-02
-1.1026E-01	-1.0907E-01	-1.0547E-01	-9.9480E-02	-9.1091E-02	-8.0307E-02	-6.7125E-02	-5.1547E-02	-3.3573E-02	-1.3201E-02
-1.1111E-01	-1.0983E-01	-1.0596E-01	-9.9522E-02	-9.0505E-02	-7.8912E-02	-6.4742E-02	-4.7997E-02	-2.8675E-02	-6.7788E-03
-1.0937E-01	-1.0799E-01	-1.0385E-01	-9.6943E-02	-8.7274E-02	-7.4844E-02	-5.9651E-02	-4.1695E-02	-2.0978E-02	0.
-1.0476E-01	-1.0328E-01	-9.8850E-02	-9.1463E-02	-8.1121E-02	-6.7823E-02	-5.1571E-02	-3.2363E-02	-1.0204E-02	0.
-9.6986E-02	-9.5410E-02	-9.0679E-02	-8.2795E-02	-7.1755E-02	-5.7562E-02	-4.0214E-02	-1.9712E-02	0.	0.
-8.5754E-02	-8.4075E-02	-7.9037E-02	-7.0638E-02	-5.8879E-02	-4.3760E-02	-2.5283E-02	-3.4433E-03	0.	0.
-7.0754E-02	-6.8970E-02	-6.3615E-02	-5.4686E-02	-4.2184E-02	-2.6110E-02	-6.4678E-03	0.	0.	0.
-5.1668E-02	-4.9775E-02	-4.4094E-02	-3.4619E-02	-2.1351E-02	-4.2919E-03	0.	0.	0.	0.
-2.8168E-02	-2.6164E-02	-2.0149E-02	-1.0110E-02	0.	0.	0.	0.	0.	0.

DECOMPOSITION INCOMPLETE ET CG
****

CPU TIME USED SO FAR : 40.4790

CPU TIME USED SO FAR : 41.6810

ITER = = 19

X =

-1.3677E-02	-1.3197E-02	-1.1758E-02	-9.3621E-03	-6.0109E-03	-1.7079E-03	0.	0.	0.	0.
-2.7230E-02	-2.6694E-02	-2.5083E-02	-2.2401E-02	-1.8649E-02	-1.3827E-02	-7.9272E-03	-9.6414E-04	0.	0.
-4.0449E-02	-3.9652E-02	-3.8060E-02	-3.5076E-02	-3.0899E-02	-2.5530E-02	-1.8966E-02	-1.1212E-02	-2.2605E-03	0.
-5.3150E-02	-5.2489E-02	-5.0507E-02	-4.7204E-02	-4.2580E-02	-3.6636E-02	-2.9370E-02	-2.0785E-02	-1.0877E-02	0.
-6.5143E-02	-6.4415E-02	-6.2232E-02	-5.8593E-02	-5.3500E-02	-4.6952E-02	-3.8949E-02	-2.9491E-02	-1.8577E-02	-6.2084E-03
-7.6226E-02	-7.5428E-02	-7.3034E-02	-6.9044E-02	-6.3458E-02	-5.6277E-02	-4.7500E-02	-3.7127E-02	-2.5158E-02	-1.1594E-02
-8.6188E-02	-8.5317E-02	-8.2702E-02	-7.8344E-02	-7.2243E-02	-6.4400E-02	-5.4813E-02	-4.3484E-02	-3.0411E-02	-1.5596E-02
-9.4809E-02	-9.3860E-02	-9.1015E-02	-8.6273E-02	-7.9635E-02	-7.1100E-02	-6.0666E-02	-4.8339E-02	-3.4114E-02	-1.7992E-02
-1.0186E-01	-1.0083E-01	-9.7743E-02	-9.2601E-02	-8.5402E-02	-7.6146E-02	-6.4833E-02	-5.1464E-02	-3.6037E-02	-1.8554E-02
-1.0709E-01	-1.0598E-01	-1.0264E-01	-9.7086E-02	-8.9304E-02	-7.9298E-02	-6.7069E-02	-5.2616E-02	-3.5940E-02	-1.7040E-02
-1.1026E-01	-1.0907E-01	-1.0547E-01	-9.9480E-02	-9.1091E-02	-8.0307E-02	-6.7125E-02	-5.1547E-02	-3.3573E-02	-1.3201E-02
-1.1111E-01	-1.0983E-01	-1.0596E-01	-9.9522E-02	-9.0505E-02	-7.8912E-02	-6.4742E-02	-4.7997E-02	-2.8675E-02	-6.7788E-03
-1.0937E-01	-1.0799E-01	-1.0385E-01	-9.6943E-02	-8.7274E-02	-7.4844E-02	-5.9651E-02	-4.1695E-02	-2.0978E-02	0.
-1.0476E-01	-1.0328E-01	-9.8850E-02	-9.1463E-02	-8.1121E-02	-6.7823E-02	-5.1571E-02	-3.2363E-02	-1.0204E-02	0.
-9.6986E-02	-9.5410E-02	-9.0679E-02	-8.2795E-02	-7.1755E-02	-5.7562E-02	-4.0214E-02	-1.9712E-02	0.	0.
-8.5754E-02	-8.4075E-02	-7.9037E-02	-7.0638E-02	-5.8879E-02	-4.3760E-02	-2.5283E-02	-3.4433E-03	0.	0.
-7.0754E-02	-6.8970E-02	-6.3615E-02	-5.4686E-02	-4.2184E-02	-2.6110E-02	-6.4678E-03	0.	0.	0.
-5.1668E-02	-4.9775E-02	-4.4094E-02	-3.4619E-02	-2.1351E-02	-4.2919E-03	0.	0.	0.	0.
-2.8168E-02	-2.6164E-02	-2.0149E-02	-1.0110E-02	0.	0.	0.	0.	0.	0.

VALEURS PROPRES DE L-1 * A * R-1 SYM.

CPU TIME USED SO FAR : 46.0170

CPU TIME USED SO FAR : 89.8070

LAMBDA =

-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00
-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00
-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	1.2066E-01	3.0555E-01	3.2319E-01
4.8923E-01	5.2044E-01	5.3091E-01	6.2907E-01	6.5823E-01	6.6541E-01	7.2474E-01	7.3594E-01	7.6679E-01	7.7568E-01
8.0087E-01	8.0899E-01	8.2696E-01	8.4058E-01	8.4913E-01	8.6023E-01	8.6499E-01	8.7730E-01	8.8138E-01	8.9261E-01
8.9734E-01	9.0119E-01	9.0433E-01	9.1055E-01	9.1575E-01	9.1911E-01	9.2179E-01	9.2702E-01	9.2771E-01	9.3007E-01
9.3228E-01	9.3474E-01	9.3675E-01	9.3820E-01	9.4009E-01	9.4254E-01	9.4360E-01	9.4539E-01	9.4671E-01	9.4773E-01
9.4931E-01	9.5015E-01	9.5061E-01	9.5093E-01	9.5215E-01	9.5350E-01	9.5467E-01	9.5612E-01	9.5631E-01	9.5715E-01
9.5746E-01	9.5866E-01	9.6107E-01	9.6291E-01	9.6320E-01	9.6397E-01	9.6676E-01	9.6797E-01	9.6930E-01	9.7109E-01
9.7407E-01	9.7470E-01	9.7665E-01	9.7697E-01	9.7789E-01	9.8023E-01	9.8287E-01	9.8330E-01	9.8488E-01	9.8600E-01
9.8921E-01	9.9080E-01	9.9185E-01	9.9236E-01	9.9996E-01	9.9998E-01	9.9999E-01	1.0000E+00	1.0000E+00	1.0000E+00
1.0000E+00	1.0000E+00	1.0001E+00	1.0078E+00	1.0083E+00	1.0096E+00	1.0111E+00	1.0139E+00	1.0167E+00	1.0181E+00
1.0191E+00	1.0200E+00	1.0245E+00	1.0253E+00	1.0257E+00	1.0280E+00	1.0287E+00	1.0297E+00	1.0325E+00	1.0349E+00
1.0363E+00	1.0406E+00	1.0426E+00	1.0447E+00	1.0453E+00	1.0469E+00	1.0490E+00	1.0514E+00	1.0538E+00	1.0549E+00
1.0553E+00	1.0554E+00	1.0579E+00	1.0605E+00	1.0612E+00	1.0626E+00	1.0653E+00	1.0675E+00	1.0683E+00	1.0690E+00
1.0696E+00	1.0708E+00	1.0720E+00	1.0734E+00	1.0757E+00	1.0766E+00	1.0797E+00	1.0815E+00	1.0827E+00	1.0872E+00
1.0895E+00	1.0933E+00	1.0940E+00	1.0968E+00	1.1024E+00	1.1054E+00	1.1123E+00	1.1159E+00	1.1180E+00	1.1253E+00
1.1301E+00	1.1353E+00	1.1398E+00	1.1479E+00	1.1564E+00	1.1639E+00	1.1664E+00	1.1777E+00	1.1880E+00	1.2010E+00
1.2051E+00	1.2160E+00	1.2318E+00	1.2444E+00	1.2515E+00	1.2753E+00	1.2775E+00	1.3008E+00	1.3145E+00	1.3354E+00

VALEURS PROPRES DE A SYMETRISEE

LAMBDA =

-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00
-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00
-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	-1.0000E+00	2.1998E+01	7.4118E+01	8.4590E+01
1.5044E+02	1.6279E+02	1.8413E+02	2.4590E+02	2.6427E+02	2.8558E+02	3.1841E+02	3.5736E+02	3.8151E+02	4.0245E+02
4.4333E+02	4.8010E+02	4.8416E+02	5.0412E+02	5.3039E+02	5.7683E+02	6.0447E+02	6.2336E+02	6.3168E+02	6.6002E+02
6.7734E+02	7.1108E+02	7.1586E+02	7.3430E+02	7.7767E+02	7.8128E+02	7.8889E+02	8.0959E+02	8.2694E+02	8.4592E+02
8.5253E+02	8.6434E+02	8.9322E+02	8.9474E+02	9.3040E+02	9.5523E+02	9.7461E+02	9.9398E+02	1.0101E+03	1.0376E+03
1.0654E+03	1.0820E+03	1.0893E+03	1.1267E+03	1.1327E+03	1.1650E+03	1.2018E+03	1.2559E+03	1.2580E+03	1.2642E+03
1.3156E+03	1.3335E+03	1.3598E+03	1.3722E+03	1.4121E+03	1.4366E+03	1.4679E+03	1.4966E+03	1.5193E+03	1.5755E+03
1.5939E+03	1.6243E+03	1.6281E+03	1.6485E+03	1.6937E+03	1.7041E+03	1.7856E+03	1.7907E+03	1.8456E+03	1.8559E+03
1.8799E+03	1.9038E+03	1.9120E+03	1.9695E+03	1.9840E+03	2.0546E+03	2.0551E+03	2.0997E+03	2.1115E+03	2.1372E+03
2.1688E+03	2.1835E+03	2.2486E+03	2.2953E+03	2.3150E+03	2.3242E+03	2.3379E+03	2.3706E+03	2.3922E+03	2.4375E+03
2.4635E+03	2.5288E+03	2.5384E+03	2.5449E+03	2.5747E+03	2.5894E+03	2.6285E+03	2.6463E+03	2.6905E+03	2.7266E+03
2.7345E+03	2.7607E+03	2.7693E+03	2.8263E+03	2.8411E+03	2.8580E+03	2.8832E+03	2.8950E+03	2.9070E+03	2.9288E+03
2.9573E+03	3.0061E+03	3.0271E+03	3.0372E+03	3.0497E+03	3.0724E+03	3.0805E+03	3.1080E+03	3.1160E+03	3.1304E+03
3.1476E+03	3.1521E+03	3.1752E+03	3.2011E+03	3.2155E+03	3.2335E+03	3.2567E+03	3.2813E+03	3.3012E+03	3.3110E+03
3.3452E+03	3.3724E+03	3.4080E+03	3.4175E+03	3.4544E+03	3.4652E+03	3.4947E+03	3.5055E+03	3.5495E+03	3.5704E+03
3.6073E+03	3.6314E+03	3.6633E+03	3.6719E+03	3.7104E+03	3.7488E+03	3.7670E+03	3.7774E+03	3.8418E+03	3.8481E+03
3.9024E+03	4.0412E+03	4.2722E+03	5.1395E+03	6.7331E+03	8.2855E+03	1.0748E+04	1.1209E+04	1.2804E+04	2.0707E+04

```
  EPF  LAUSANNE NOS/BE 1.3 LEV 486-17 09/11/82
09.49.43.T255175  FROM
09.49.43.IP   00000192 WORDS - FILE INPUT  , DC 04
09.49.43.T2551,T1000,CM240000.      GRUEER,CRPP
09.49.49.ROUTE,OUTPUT,DEF,TID=C,REP=25.
09.49.49.ATTACH,TAPE12,MAT2010N,ID=L18RS,MR=1.
09.49.49.PF CYCLE NO. = 001
09.49.49.ATTACH,LINPACK.
09.49.49.PFN IS
09.49.49.LINPACK
09.49.50.PF CYCLE NO. = 001
09.49.50.ATTACH,EISPACK.
09.49.50.PFN IS
09.49.50.EISPACK
09.49.50.PF CYCLE NO. = 100
09.49.50.ATTACH,NAGF.
09.49.50.PFN IS
09.49.50.NAGF
09.49.50.PF CYCLE NO. = 001
09.49.50.ATTACH,OLDPL,MATUPD,ID=L18P1,MR=1.
09.49.50.PF CYCLE NO. = 003
09.49.51.UPDATE,Q,L=7.
09.50.11.  UPDATE COMPLETE.
09.50.11.FTN,I=COMPILE,OPT=2,L=0.
09.56.32.     15.298 CP SECONDS COMPILATION TIME
09.56.32.MAP(PART)
09.56.32.LIBRARY,LINPACK,EISPACK.
09.56.32.LDSET,LIB=NAGF.
09.56.32.LGO.
10.33.07.LOCKIN.
10.33.58.     STOP
10.33.58.211500B CHAMP MEMOIRE MAXIMUM
10.33.58.    113.255 CP SECONDS EXECUTION TIME
10.33.58.OP   00018240 WORDS - FILE OUTPUT , DC 40
10.33.58.MS    21504  WORDS  (   78848 MAX USEC)
10.33.58.CPA       55.094 SEC.          1.836 ADJ.
10.33.58.CPB       79.250 SEC.          2.641 ADJ.
10.33.58.IO        20.149 SEC.          1.343 ADJ.
10.33.58.CM      8965.186 KWS.        291.835 ADJ.
10.33.58.SS*** COUT DU CALCUL        208.80 FR ***
10.33.59.CO PERIPHERIE EN SUS (MESS CC, TRACEUR)
10.33.58.PP        80.920 SEC.     DATE 30/11/82
10.33.58.CC      148.30 FR I/O    0 PAUSES
10.33.58.EJ  END OF JOB, **
```

Appendice 3C Programme MUGRID
==========

Le problème $\underline{A}\underline{x} = \underline{b}$ est résolu par une variante de la méthode à réseaux multiples. La description au début du code montre la modularité de MUGRID.

```
*DECK MUGRID
      PROGRAM MUGRID(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE10)
C
C                          ********************
C                          *                  *
C                          * PROGRAM MUGRID   *
C                          *                  *
C                          ********************
C
C                               **********
C
C     USING MULTIGRID COMPUTE SOLOVEV EQUILIBRIA IN A BOX
C     WHERE THE ANALYTICAL PSI VALUES ARE PRESCRIBTED ON THE FRAME
C                               **********
C
C     TOP - BOTTOM SYMMETRY  ===>  ONLY THE UPPER HALF OF THE DOMAIN IS
C     CONSIDERED AND SYMMETRY IS IMPOSED BY SETTING EQUAL PSI VALUES ON
C     BOTH SIDES OF THE HORIZONTAL AXIS
C                               **********
C
C     THIS VERSION ALLOWS :
C                   - MAXIMUM NUMBERS OF INTERVALS NR=126 AND NZ=64
C                     (NZ MUST BE EQUAL TO NR/2 ; BOTH POWER OF 2)
C                   - A MAXIMUM MULTIGRID LEVELS NUMBER NIVEAU=5
C                   - A MAXIMUM "MENU"-STEPS NUMBER NM=20
C                               **********
C
C     DISK CHANNELS :
C     ---------------
C
C     TAPE  5       INPUT
C     TAPE  6       OUTPUT
C     TAPE 10       AUXILLIARY STORAGE
C                               **********
C
C     LIST OF SUBPROGRAMS :
C     ---------------------
C
C     MASTER        CONTROL THE RUN                                 0.01
C
C     LABRUN        LABEL THE RUN                                   1.01
C     CLEAR         CLEAR ALL COMMONS                               1.02
C     PRESET        SET UP THE DEFAULT CASE                         1.03
C     DATA          READ IN NAMELIST                                1.04
C     AUXVAL        SET UP AUXILLIARY VALUES                        1.05
C     COTROL        CONTROL READ IN PARAMETERS                      1.06
C
C     STEPON        LEAD THE CALCULATIONS                           2.01
C     MESH          CREATE FRAME AND MESH                           2.04
```

```
C     FRAME          PRESCRIBE ANALYTICAL SOLUTION                  2.07
C     GUESS          COMPUTE INITIAL PSI SOLUTION                   2.10
C     RSTART         READ OLD EQUILIBRIUM                           2.13
C     MAGAXE         FIND MAGNETIC AXIS                             2.16
C     TEST           COMPUTE RELATIVE ERROR                         2.19
C     SOLVIT         SOLVE GRAD-SHAFRANOV EQUATION                  2.22
C     CHANGE(2)      MODIFY THE "MENU"                              2.25
C     CHOOSE(1)      REDUCE,KEEP OR EXPAND THE MESH                 2.28
C     SPS(2)         COMPUTE PSI(R,Z)                               2.31
C     SOURCE(1)      COMPUTE SOURCE TERM                            2.34
C     REDUCE(2)      DIVIDE MESH SIZE                               2.37
C     EXTPSI(1)      EXTRAPOLATE FROM MULTIGRID LEVEL               2.40
C     EXPAND(1)      DOUBLE THE MESH SIZE                           2.43
C     CUBIC(3)       CUBIC INTERPOLATION                            2.46
C
C     OUTPUT(1)      CONTROL INPUT/OUTPUT                           3.01
C     IODISK(1)      PERFORM DISK OPERATIONS                        3.02
C     STOGET(2)      STORE OR GET CONVERGED SOLUTIONS               3.05
C
C                          **********
C
C     AUXILLIARY SUBROUTINES :
C     ------------------------
C
C     SAXPY          Y = Y + A * X                                  MAT1
C     SCOPY          Y = X                                          MAT2
C     SDOT           SDOT = X . Y                                   MAT3
C     SSCAL          X = A * X                                      MAT4
C     SXYPZ          Z = Z + X . Y                                  MAT5
C     VZERO          X = 0                                          MAT6
C
C     PAGE           JUMP A PAGE                                    OLYMP
C     BLINES(1)      JUMP N LINES                                   OLYMP
C     MESAGE(1)      WRITE A 48-CHARACTERS STRING                   OLYMP
C     IVAR(2)        WRITE NAME AND INTEGER VALUE                   OLYMP
C     RVAR(2)        WRITE NAME AND REAL VALUE                      OLYMP
C     IARRAY(3)      WRITE NAME AND INTEGER ARRAY                   OLYMP
C     RARRAY(3)      WRITE NAME AND REAL ARRAY                      OLYMP
C     RESETI(3)      RESET INTEGER ARRAY                            OLYMP
C     RESETR(3)      RESET REAL ARRAY                               OLYMP
C
C                          **********
C
C     LIST OF COMMONS :
C     -----------------
C
C     COMBLA         PSI AND SOURCE ARRAYS                          C.01
C     COMCON         CONTROL PARAMETERS                             C.02
C     COMESH         MESH QUANTITIES                                C.03
C     COMIOD         INPUT/OUTPUT                                   C.04
C     COMLAB         LABEL OF THE RUN                               C.05
C     COMMGD         MULTIGRID PARAMETERS                           C.06
C     COMNUM         MESH NUMBERS                                   C.07
C     COMSOL         RESOLUTION PARAMETERS                          C.08
C
C     COMDAT         NAMELIST                                       C.20
```

```
C
C                         **********
C
C     LIST OF VARIABLES :
C     -------------------
C
C     CPSI1       PSI SOLUTION NR= 8 NZ= 4              RA    C.01
C     CPSI2       PSI SOLUTION NR=16 NZ= 8              RA    C.01
C     CPSI3       PSI SOLUTION NR=32 NZ=16              RA    C.01
C     CPSI4       PSI SOLUTION NR=64 NZ=32              RA    C.01
C     CPSI        WORKING PSI ARRAY                     RA    C.01
C     CPSO        AUXILLIARY PSI ARRAY                  RA    C.01
C     SSS         SOURCE ARRAY                          RA    C.01
C
C     NOPT        GUESS OR RSTART OPTION                I     C.02
C     NLCHAN      MODIFY OR NOT "MENU"-LIST             L     C.02
C     NLEXTR      LEVEL EXTRAPOLATION OR NOT            L     C.02
C     NLJUMP      JUMP RESTART IF MESH SIZE KEPT        L     C.02
C     NLTEST      COMPUTE RELATIVE ERROR                L     C.02
C
C     ASPCT       ASPECT RATIO                          R     C.03
C     CONST       DEFINE OVERRELAXATION PARAMETER       R     C.03
C     ELLIPT      ELLIPTICITY                           R     C.03
C     HEIGHT      HALF HEIGHT OF THE FRAME              R     C.03
C     R0          PLASMA GEOMETRIC CENTER               R     C.03
C     RC          PRESCRIBED PLASMA MAGNETIC AXIS       R     C.03
C     RMAG        COMPUTED MAGNETIC AXIS                R     C.03
C     RR          MESH POSITION IN R                    RA    C.03
C     RZ          MESH POSITION IN Z                    RA    C.03
C     WIDTH       HALF WIDTH OF THE FRAME               R     C.03
C
C     NMAX        AUXILLIARY VALUE FOR I/O              I     C.04
C
C     LABEL1      READ IN COMMENT CARD                  IA    C.05
C     LABEL2      READ IN COMMENT CARD                  IA    C.05
C     LABEL3      READ IN COMMENT CARD                  IA    C.05
C     LABEL4      READ IN COMMENT CARD                  IA    C.05
C
C     CALPHA      LIST OF OVERRELAXATION PARAMETERS     R     C.06
C     NIVEAU      LEVELS NUMBER                         I     C.06
C     NM          STEPS NUMBER                          I     C.06
C     NMMAX       MAXIMUM LEVELS NUMBER                 I     C.06
C     NMENU       "MENU"-LIST                           IA    C.06
C     NNINSC      LIST OF EXTERNAL ITERATIONS NUMBER    IA    C.06
C     NNITER      LIST OF INTERNAL ITERATIONS NUMBER    IA    C.06
C     NNR         LIST OF INTERVALS NUMBER IN R         IA    C.06
C     NNZ         LIST OF INTERVALS NUMBER IN Z         IA    C.06
C     NRATIO      CONTROL STORAGE AND PICK UP           I     C.06
C     NTEP        MAXIMUM STEPS NUMBER                  I     C.06
C
C     MRMAX       MAX INTERVALS NUMBER IN R             I     C.07
C     MRMIN       MIN INTERVALS NUMBER IN R             I     C.07
C     MRZMAX      MESH POINTS NUMBER FOR PICK UP        I     C.07
C     MZMAX       MAX INTERVALS NUMBER IN Z             I     C.07
C     NBRMAX      RIGHT LIMIT OF MESH POINTS IN R       I     C.07
C     NBRMIN      LEFT LIMIT OF MESH POINTS IN R        I     C.07
```

```
C     NBZMAX      UPPER LIMIT OF MESH POINTS IN Z          I     C.07
C     NITER       MAX NUMBER OF INTERNAL ITERATIONS        I     C.07
C     NINSCA      MAX NUMBER OF EXTERNAL ITERATIONS        I     C.07
C     NR          INTERVALS NUMBER IN R                    I     C.07
C     NR1         MESH POINTS NUMBER IN R                  I     C.07
C     NZ          INTERVALS NUMBER IN Z                    I     C.07
C     NZ1         MESH POINTS NUMBER IN Z                  I     C.07
C
C     ALPHA       OVERRELATION PARAMETER                   R     C.08
C     CB2         AUXILLIARY SOLOVEV TERM                  R     C.08
C     CPP         DP/DPSI                                  R     C.08
C     CQ0         Q0                                       R     C.08
C     CTTP        T*DT/DPSI                                R     C.08
C     EPSCON      CONVERGENCE LIMIT                        R     C.08
C     DRAG        FOR CONVERGENCE PROBLEM                  R     C.08
C     RESIDU      MEASURE OF THE SOLUTION ACCURACY         R     C.08
C     SPSI0       PRESCRIBED PSI MINIMUM                   R     C.08
C     SPSIM       COMPUTED PSI MINIMUM                     R     C.08
C
C                          **********
C
C     NAMELIST VARIABLES :
C     --------------------
C
C     ASPCT       COMESH          BASIC          R     C.03
C     CALPHA      COMMGD          BASIC          R     C.06
C     CB2         COMSOL                         R     C.08
C     CONST       COMESH          BASIC          R     C.03
C     CPP         COMSOL                         R     C.08
C     CQ0         COMSOL          BASIC          R     C.08
C     CTTP        COMSOL          BASIC          R     C.08
C     DRAG        COMESH          BASIC          R     C.03
C     ELLIPT      COMESH          BASIC          R     C.03
C     EPSCON      COMSOL          BASIC          R     C.08
C     HEIGHT      COMESH                         R     C.03
C     R0          COMESH                         R     C.03
C     RC          COMESH          BASIC          R     C.03
C     SPSI0       COMSOL                         R     C.08
C     WIDTH       COMESH                         R     C.03
C     NIVEAU      COMMGD          BASIC          I     C.06
C     NM          COMMGD          BASIC          I     C.06
C     NMENU       COMMGD          BASIC          IA    C.06
C     NNINSC      COMMGD          BASIC          IA    C.06
C     NNITER      COMMGD          BASIC          IA    C.06
C     NR          COMESH          BASIC          I     C.03
C     NRATIO      COMMGD                         I     C.06
C     NZ          COMESH          BASIC          I     C.03
C     NLCHAN      COMCON          BASIC          L     C.02
C     NLEXTR      COMCON          BASIC          L     C.02
C     NLTEST      COMCON          BASIC          L     C.02
C
C                          **********
C
C     DEFAULT CASE :
C     --------------
C
```

```
C     ASPCT=1/3  CQ0=1  CTTP=0  ELLIPT=2  RC=1
C     NIVEAU=4  NM=15  NNINSC=15 10 5 3  NNITER=4*10  NR=64  NZ=32
C     NLEXTR=.T.  NLTEST=.F.
C
C     RESULT : PSIMIN=-1/9  MAG AXIS=.99994  IN 5 EXTERNAL ITERATIONS
C     ---------
C
C                              **********
C
C     NOTICE THAT 4 COMMENT CARDS MUST ALWAYS PRECEED THE NAMELIST
C     ------ IN THE INPUT RECORD
C
C
C  *    *    *    *    *    *    *    *    *    *    *    *    *    *
C
C
C     START THE RUN
C
      CALL MASTER
C
      STOP 0000
      END
0        1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

SOLOVEV EQUILIBRIUM IN A BOX WITH PRESCRIBED
ANALYTICAL PSI VALUES ON THE FRAME

*** TEST MUGRID :
***
*** NR=128 NZ=64 NLEXTR=.T.
*** LEVELS NUMBER : NIVEAU=5

```
$MGDDAT
ASPCT   = .33333333333333E+00,
CALPHA  = .145E+01, .17E+01, .18E+01, .1E+01, .1E+01,
CB2     = 0.0,
CPP     = .25E+01,
CQQ     = .1E+01,
CTTP    = 0.0,
DRAG    = 0.0,
ELLIPT  = .2E+01,
EPSCON  = .1E-07,
HEIGHT  = .71374417954618E+00,
R0      = .93417235896271E+00,
RC      = .1E+01,
SPSI0   = .11111111111111E+00,
WIDTH   = .35692208977309E+00,
NIVEAU  = 5,
NM      = 15,
NMENU   = 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
NNINSC  = 15, 10, 5, 3, 2,
NNITER  = 10, 10, 10, 10, 10,
NR      = 128,
NRATIO  = 0,
NZ      = 64,
NLCHAN  = T,
NLEXTR  = T,
NLTEST  = F,
$END

JM =   1   NIVEAU =   1   NR =    8   NZ =    4   NINSCA =  15   NITER =   10   ALPHA =    1.4500E+00
         PSIMIN =   -1.0000E+00    RMAG =    9.3417E-01
         PSIMIN =   -1.1272E-01    RMAG =    9.9369E-01      JIT =  11    RESIDU =  .0009411133    EPSCON =  .0000000100
         PSIMIN =   -1.1152E-01    RMAG =    9.9696E-01      JIT =  11    RESIDU =  .0000007349    EPSCON =  .0000000100
                                                             JIT =   8    RESIDU =  .0000000042    EPSCON =  .0000000100
```

```
JM =   2   NIVEAU =   2   NR =   16   NZ =    8   NINSCA =  10   NITER =   10   ALPHA =    1.7000E+00
         PSIMIN =   -1.1131E-01    RMAG =    9.9948E-01          JIT =  11   RESIDU =  .0000154501   EPSCON =  .0000000100
         PSIMIN =   -1.1115E-01    RMAG =    9.9960E-01          JIT =  11   RESIDU =  .0000004091   EPSCON =  .0000000100
         PSIMIN =   -1.1113E-01    RMAG =    9.9964E-01          JIT =  11   RESIDU =  .0000000132   EPSCON =  .0000000100
         PSIMIN =   -1.1113E-01    RMAG =    9.9964E-01          JIT =   1   RESIDU =  .0000000097   EPSCON =  .0000000100

JM =   3   NIVEAU =   3   NR =   32   NZ =   16   NINSCA =   5   NITER =   10   ALPHA =    1.8000E+00
         PSIMIN =   -1.1113E-01    RMAG =    9.9974E-01          JIT =  11   RESIDU =  .0000006391   EPSCON =  .0000000100
         PSIMIN =   -1.1113E-01    RMAG =    9.9975E-01          JIT =  11   RESIDU =  .0000001174   EPSCON =  .0000000100
         PSIMIN =   -1.1113E-01    RMAG =    9.9975E-01          JIT =  11   RESIDU =  .0000000268   EPSCON =  .0000000100
         PSIMIN =   -1.1113E-01    RMAG =    9.9975E-01          JIT =   7   RESIDU =  .0000000081   EPSCON =  .0000000100

JM =   4   NIVEAU =   4   NR =   64   NZ =   32   NINSCA =   3   NITER =   10   ALPHA =    1.0000E+00
         PSIMIN =   -1.1111E-01    RMAG =    9.9994E-01          JIT =  11   RESIDU =  .0000000101   EPSCON =  .0000000100
         PSIMIN =   -1.1111E-01    RMAG =    9.9994E-01          JIT =   1   RESIDU =  .0000000090   EPSCON =  .0000000100

JM =   5   NIVEAU =   5   NR =  128   NZ =   64   NINSCA =   2   NITER =   10   ALPHA =    1.0000E+00
         PSIMIN =   -1.1111E-01    RMAG =    9.9999E-01          JIT =   2   RESIDU =  .0000000093   EPSCON =  .0000000100

MAG AXIS =    9.9999E-01
PSI MIN  =   -1.1111E-01
TEST PSM =    6.6673E-06

SOL Z=0
   4.8851E-05  -2.0974E-03  -4.2431E-03  -6.3876E-03  -8.5303E-03  -1.0670E-02  -1.2808E-02  -1.4941E-02  -1.7070E-02  -1.9194E-02
  -2.1313E-02  -2.3425E-02  -2.5530E-02  -2.7628E-02  -2.9717E-02  -3.1798E-02  -3.3868E-02  -3.5929E-02  -3.7978E-02  -4.0016E-02
  -4.2041E-02  -4.4053E-02  -4.6051E-02  -4.8034E-02  -5.0002E-02  -5.1954E-02  -5.3889E-02  -5.5807E-02  -5.7706E-02  -5.9587E-02
  -6.1447E-02  -6.3287E-02  -6.5105E-02  -6.6901E-02  -6.8674E-02  -7.0424E-02  -7.2149E-02  -7.3848E-02  -7.5521E-02  -7.7167E-02
  -7.8786E-02  -8.0376E-02  -8.1936E-02  -8.3466E-02  -8.4964E-02  -8.6431E-02  -8.7865E-02  -8.9265E-02  -9.0630E-02  -9.1960E-02
  -9.3253E-02  -9.4510E-02  -9.5728E-02  -9.6906E-02  -9.8045E-02  -9.9143E-02  -1.0020E-01  -1.0121E-01  -1.0218E-01  -1.0311E-01
  -1.0399E-01  -1.0482E-01  -1.0561E-01  -1.0634E-01  -1.0703E-01  -1.0767E-01  -1.0826E-01  -1.0879E-01  -1.0927E-01  -1.0970E-01
  -1.1007E-01  -1.1038E-01  -1.1064E-01  -1.1084E-01  -1.1098E-01  -1.1106E-01  -1.1108E-01  -1.1104E-01  -1.1093E-01  -1.1076E-01
  -1.1052E-01  -1.1021E-01  -1.0984E-01  -1.0940E-01  -1.0869E-01  -1.0831E-01  -1.0766E-01  -1.0693E-01  -1.0613E-01  -1.0525E-01
  -1.0430E-01  -1.0327E-01  -1.0216E-01  -1.0097E-01  -9.9704E-02  -9.8353E-02  -9.6918E-02  -9.5399E-02  -9.3794E-02  -9.2102E-02
  -9.0323E-02  -8.8454E-02  -8.6495E-02  -8.4445E-02  -8.2302E-02  -8.0065E-02  -7.7733E-02  -7.5305E-02  -7.2779E-02  -7.0155E-02
  -6.7430E-02  -6.4605E-02  -6.1577E-02  -5.8645E-02  -5.5508E-02  -5.2266E-02  -4.8915E-02  -4.5456E-02  -4.1887E-02  -3.8207E-02
  -3.4414E-02  -3.0508E-02  -2.6486E-02  -2.2347E-02  -1.8091E-02  -1.3716E-02  -9.2201E-03  -4.6026E-03   1.3792E-04
```

CHAPITRE VI LA METHODE DES ELEMENTS FINIS

6.1 INTRODUCTION

Dans le chapitre III, nous avons résolu l'exemple d'application de l'équilibre MHD par intermédiaire des différences finies. Par une telle discrétisation, l'opérateur, a priori symétrique et défini positif, a été rendu non symétrique. L'opérateur ne redevient symétrique que pour une discrétisation infiniment fine. La perte de la symétrie et comme conséquence, la perte d'efficacité dans la méthode de résolution, pourrait être évitée. Une méthode qui assure la symétrie du problème discrétisé est la méthode de Galerkin. L'inconnue est représentée par une combinaison linéaire de fonctions linéairement indépendantes et les équations différentielles sont mises sous forme intégrale.

Une forme intégrale particulière est la forme variationnelle que l'on rencontre souvent en physique. La solution cherchée est alors identique au minimum de cette forme. Cette formulation demande que l'opérateur soit défini positif. Cela revient à demander que A soit définie positive si l'on remplace la résolution d'un système d'équations linéaires

$$A\underline{x} = \underline{b} \tag{6.1}$$

par le problème: Trouver le minimum de

$$W(x) = 1/2(\underline{x},A\underline{x}) - (\underline{x},\underline{b}). \tag{6.2}$$

Une formulation plus générale est la formulation dite faible (ou de Galerkin) qui est applicable non seulement à des opérateurs symétriques et définis positifs mais à n'importe quel opérateur. La solution ne constitue pas forcément un minimum, mais un point stationnaire.

L'introduction de la méthode des éléments finis et le traitement d'une non-linéarité sont présentés par intermédiaire d'exemples uni-dimensionnels. Aussi discutons-nous des approches avec des éléments d'ordre plus élevé (éléments de Lagrange et de Hermite).

Nous allons après reprendre l'exemple d'application du chapitre III de l'équilibre MHD avec un membre de droite constant et traitable analy-

tiquement. Les éléments finis deux-dimensionnels sont introduits en considérant un domaine rectangulaire.

Pour finir, nous allons traiter le cas réellement appliqué, le domaine n'est plus rectangulaire, mais courbe et le membre de droite est non linéaire. Les éléments isoparamétriques sont introduits et des méthodes pour traiter la non-linéarité sont discutées.

6.2 METHODE DES ELEMENTS FINIS

6.2.1 FORMULATION FAIBLE (OU DE GALERKIN)

Considérons comme exemple l'équation

$$y''(x) = F(x), \qquad 0 < x < 1 \tag{6.3}$$

et

$$y(0) = 0 \tag{6.4}$$

$$y'(1) = 0.$$

On peut multiplier l'équation (6.3) par une fonction de test $\eta(x)$ et intégrer sur tout le domaine. La formulation dite forte est: "Chercher $y(x)$ suffisamment régulière avec $y(0) = 0$ et $y'(1) = 0$ tel que

$$\int_0^1 \eta \left[y'' - F \right] dx = 0 \tag{6.5}$$

pour tout $\eta \in L^2(0,1)$[1]".

Introduisons l'espace V qui est l'ensemble des fonctions $f(x) \in L^2(0,1)$, $f'(x) \in L^2(0,1)$ et $f(0) = 0$. En intégrant (6.5) par partie on obtient la formulation dite faible ou variationnelle ou de Galerkin:

---

[1] $L^2(0,1)$ est l'ensemble de toutes les fonctions $f(x)$ carrées intégrables dans le domaine $x \in (0,1)$.

"Chercher $y \in V$ tel que

$$\int_0^1 (\eta' y' + \eta F)dx = 0 \qquad (6.6$$

pour toute $\eta \in V$".

La formulation (6.6) a strictement la même solution que (6.3) et (6.4). Elle est valable pour un opérateur quelconque. Notons que lors de l'intégration par partie, la contribution en $x = 0$ tombe puisque $\eta = 0$ et celle en $x = 1$ s'annule puisque $y' = 0$. Dans le problème (6.6) la condition $y'(1) = 0$ est devenue une condition naturelle et ne doit plus être imposée.

Vu son application générale, nous utiliserons la formulation (6.6) dans tout le chapitre VI. Il faut noter que l'intégrale (6.6) ne contient que des dérivées premières. Dans une représentation fonctionnelle de l'inconnue y, on peut choisir des fonctions qui sont dérivables au moins une fois. Pour résoudre (6.6), la fonction y est développée en

$$y(x) = \sum_{i=1}^{N} y_i e_i(x) \qquad (6.7)$$

où les y_i sont des coéfficients inconnus et les $e_i(x)$ des fonctions linéairement indépendantes à choisir. Dans le calcul de structures d'atomes ou de molécules, les $e_i(x)$ sont des fonctions d'ondes globales en général non nulles sur tout le domaine. Dans le cas des éléments finis, les $e_i(x)$ sont des fonctions locales, non nulles sur une région restreinte uniquement.

6.6.2 ELEMENTS FINIS LINEAIRES

Discrétisons d'abord le domaine $0 \leq x \leq 1$ en N intervalles non forcément équidistants (voir figure 6.1). La solution $y(x)$, la plus simple décrite par eq. (6.7), est composée de morceaux linéaires dans chaque intervalle $x_{j-1} \leq x \leq x_j$, $j = 1, \ldots, N$ (voir figure 6.2). A chaque inconnue est attribuée une fonction de base

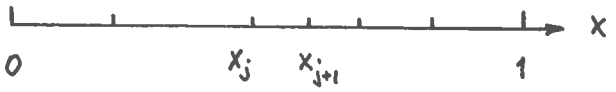


Fig. 6.1 : Discretisation du domaine $0 \leq x \leq 1$ en N = 6 intervalles non équidistants

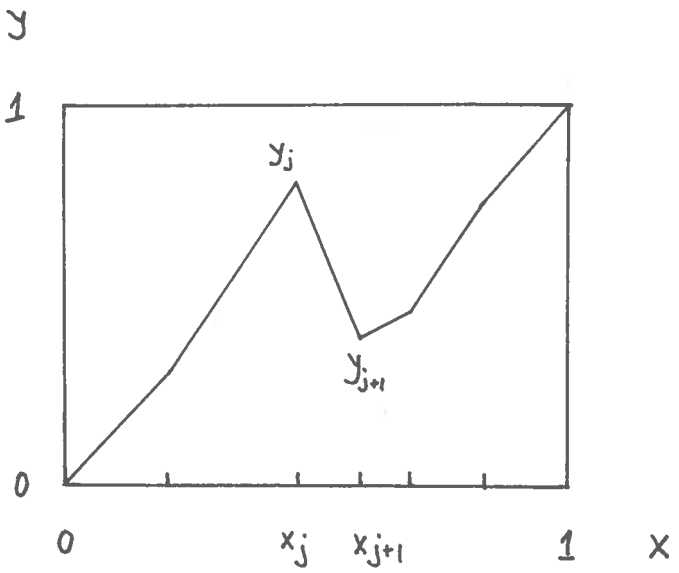


Fig. 6.2 : Représentation de y(x) par une ligne brisée (linéaire par morceau).

$$e_j(x) = \begin{cases} 0 & , \quad 0 \leq x < x_{j-1} \\ \dfrac{x-x_{j-1}}{x_j-x_{j-1}} & , \quad x_{j-1} \leq x \leq x_j \\ \dfrac{x-x_{j+1}}{x_j-x_{j+1}} & , \quad x_j \leq x \leq x_{j+1} \\ 0 & , \quad x_{j+1} < x \leq 1 \end{cases} \tag{6.8}$$

qui vaut 1 à l'endroit $x = x_j$ décroît linéairement des deux côtés, atteint zéro en $x = x_{j-1}$ et $x = x_{j+1}$ et vaut zéro pour $0 \leq x \leq x_{j-1}$ et $x_{j+1} \leq x \leq 1$. Cet elément de base est représenté dans fig. 6.3. La fonction $y(x)$ dans l'intervalle $x_j \leq x \leq x_{j+1}$ est donc représentée par (voir figures 6.2 et 6.4)

$$\begin{aligned} y(x) &= y_j\, e_j(x) + y_{j+1}\, e_{j+1}(x) \\ &= y_j \frac{x-x_{j+1}}{x_j-x_{j+1}} + y_{j+1} \frac{x-x_j}{x_{j+1}-x_j} \quad . \end{aligned} \tag{6.9}$$

En choisissant N intervalles, il y a N points du réseau et N inconnues (eq. 6.7). Il faut noter que la condition de Dirichlet $y(0) = 0$ a déjà été introduite et que la condition de Neumann $y'(1) = 0$ est une condition naturelle. Afin d'obtenir N équations nécessaire pour déterminer ces N inconnues, il faut choisir N fonctions $\eta(x)$ linéairement indépendantes.
On choisit en général

$$\eta_i(x) \equiv e_i(x), \quad i = 1,\ldots, N. \tag{6.10}$$

Cela garantit que la matrice A du système $A\underline{x} = \underline{b}$ soit symétrique. L'équation (6.6) s'écrit alors

$$\sum_{j=1}^{N} y_j \int_0^1 e_i' e_j' \, dx + \int_0^1 e_i F \, dx = 0, \quad i = 1,\ldots, N. \tag{6.11}$$

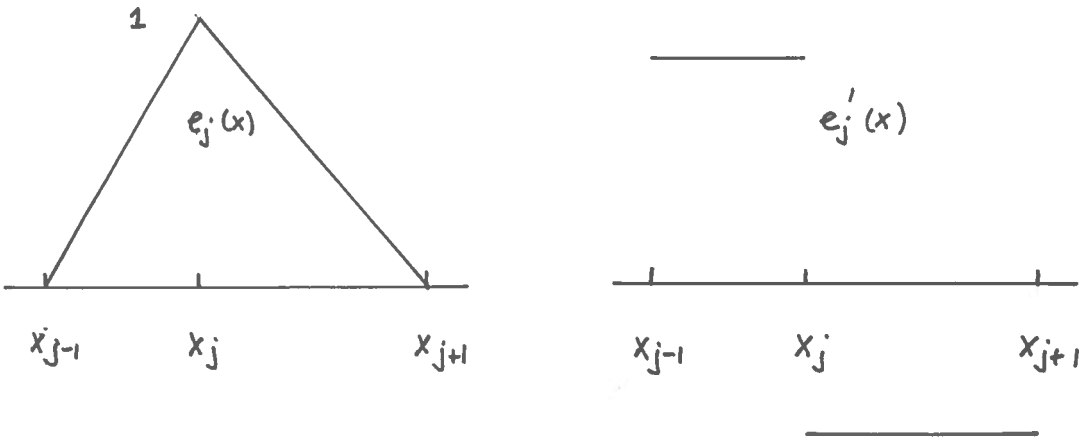


Fig. 6.3 : Fonction de base $e_j(x)$ qui atteint la valeur 1 à $x = x_j$ et sa dérivée qui est discontinue.

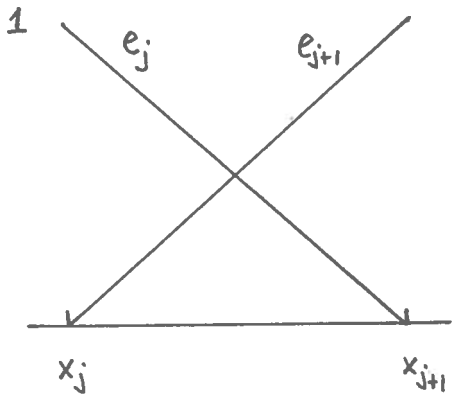


Fig. 6.4 : Les deux seules fonctions de base $e_j(x)$ et $e_{j+1}(x)$ non nulles dans l'intervalle $x_j \leq x \leq x_{j+1}$.

Le système d'équations linéaires (6.9) peut être écrit sous forme matricielle:

$$\sum_{j=1}^{N} a_{ij}\, y_j + b_i = 0 \qquad i = 1,\dots, N \tag{6.12}$$

où $$a_{ij} = \int_{x_{i-1}}^{\min(x_{i+1},x_N)} e_i'\, e_j'\,dx \tag{6.13}$$

$$b_i = \int_{x_{i-1}}^{\min(x_{i+1},x_N)} e_i\, F\,dx. \tag{6.14}$$

Il faut noter que l'intégration dans (6.13) et (6.14) ne se fait que sur deux intervalles au maximum. Cela est dû au fait que la fonction de test $\eta_i = e_i$ n'est non nulle que sur $x_{i-1} \leqslant x \leqslant x_{i+1}$.

En pratique, la construction de la matrice $A = \{a_{ij}\}$ et du membre de droite $b = \{b_i\}$ se fait en sommant sur tous les intervalles. Par intervalle il y a quatre contributions non nulles pour A et deux contributions pour $\underline{b}$. Pour l'intervalle $x_i \leqslant x \leqslant x_{j+1}$ et les éléments définis dans (6.8) on effectue les opérations suivantes pour A:

$$\begin{aligned}
a_{ii} &= a_{ii} + \int_{x_i}^{x_{i+1}} \frac{dx}{(x_{i+1} - x_i)^2} \\
a_{i\ i+1} &= a_{i\ i+1} - \int_{x_i}^{x_{i+1}} \frac{dx}{(x_{i+1} - x_i)^2} \qquad (6.15) \\
a_{i+1\ i+1} &= a_{i+1\ i+1} + \int_{x_i}^{x_{i+1}} \frac{dx}{(x_{i+1} - x_i)^2}
\end{aligned}$$

et pour $\underline{b}$:

$$b_i = b_i + \int_{x_i}^{x_{i+1}} \frac{x_{i+1} - x}{x_{i+1} - x_i} f dx$$

$$b_{i+1} = b_{i+1} + \int_{x_i}^{x_{i+1}} \frac{x - x_i}{x_{i+1} - x_i} f dx \qquad (6.16)$$

Ces intégrales sont calculés numériquement. Il faut choisir une méthode d'intégration numérique dite consistante impliquant l'obtention de la solution exacte si tous les coefficients sont constants. Pour des fonctions de base $e_j(x)$ linéaires, la méthode de Simpson ou la méthode de Gauss à deux points (voir Chapitre 2) est adéquate. Pour un grillage équidistant, l'approximation de l'opérateur y" par la méthode des éléments finis donne, à un facteur prêt, la même matrice A que par la méthode des différences finies.

La loi de convergence d'une méthode d'éléments finis linéaires est $O(h^2)$ où h est la taille de l'intervalle le plus grand. Cette loi est valable aussi bien pour un grillage équidistant que non équidistant.

6.2.3 ELEMENTS DE LAGRANGE

6.2.3.1 Fonction de Lagrange d'ordre 2

Le développement (6.7) de la fonction inconnue peut se faire avec des fonctions de base d'ordre deux, ce qui donne un y parabolique dans chaque intervalle. Une parabole est déterminée par trois points distincts. Il faut donc introduire une nouvelle variable dans chaque intervalle que l'on place au milieu. Par intervalle $x_j \leqslant x \leqslant x_{j+1}$ on a les trois fonctions de base f_j, $f_{j+1/2}$ et f_{j+1} non nulles (voir figure 6.5). La variable y dans cet intervalle s'écrit:

$$y(x_j \leqslant x \leqslant x_{j+1}) = y_j f_j(x) + y_{j+1/2} f_{j+1/2}(x) + y_{j+1} f_{j+1}(x). \qquad (6.17)$$

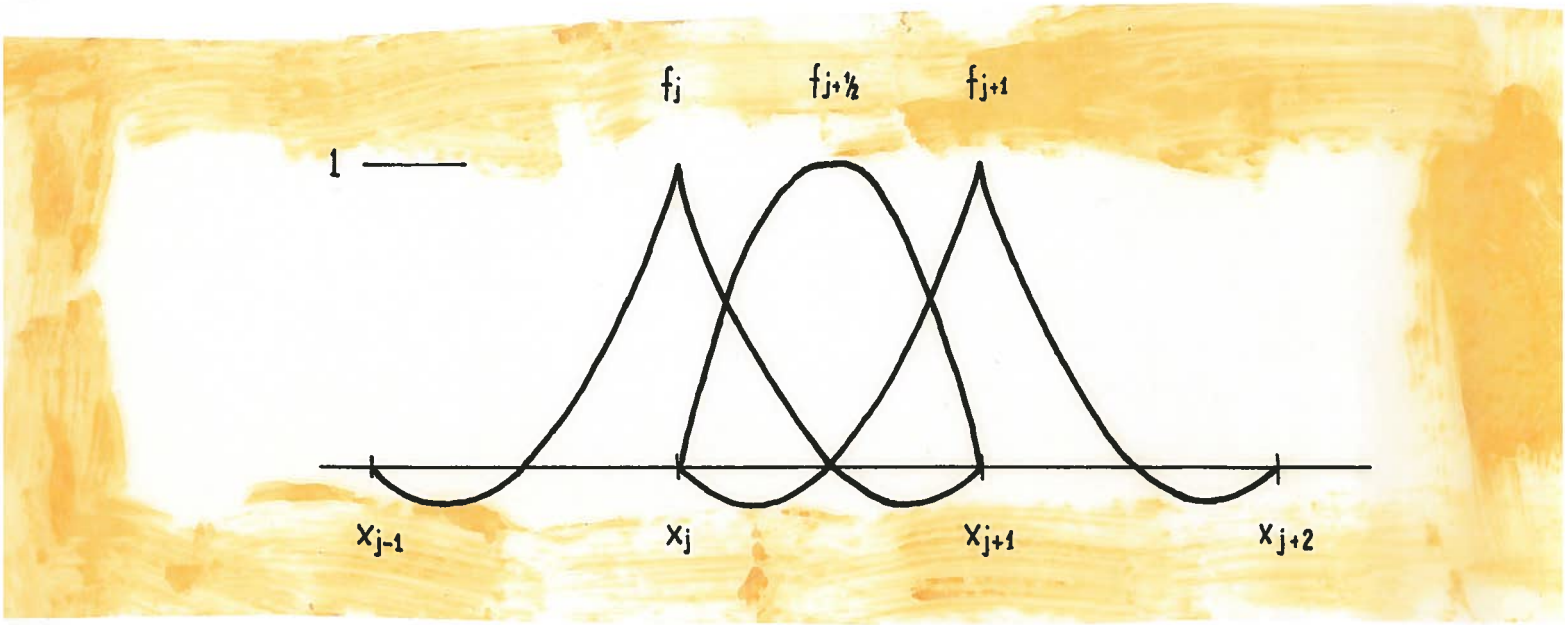


Fig. 6.5 : Les trois éléments de Lagrange d'ordre 2 non nuls dans l'intervalle $x_j \leq x \leq x_{j+1}$. L'élément $f_{j+1/2}(x)$ est nul à l'extérieur de cet intervalle.

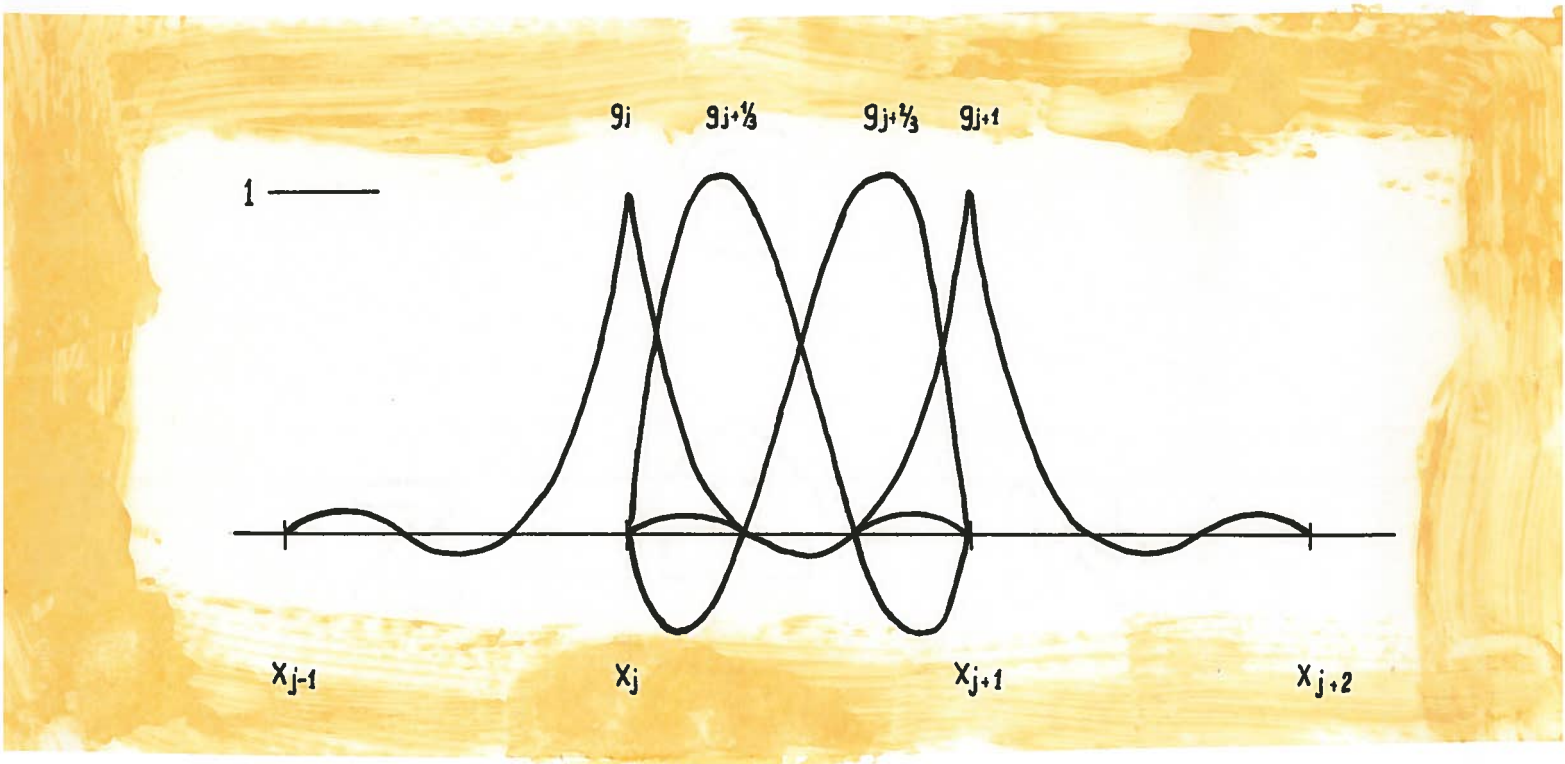


Fig. 6.6 : Les quatre éléments de Lagrange d'ordre 3 non nuls dans l'intervalle $x_j \leq x \leq x_{j+1}$. Les éléments $g_{j+1/3}$ et $g_{j+2/3}$ sont nuls à l'extérieur de l'intervalle.

Introduisons les formes fonctionnelles

$$y = y_j \frac{(x-x_{j+1/2})(x-x_{j+1})}{(x_j-x_{j+1/2})(x_j-x_{j+1})} + y_{j+1/2} \frac{(x-x_j)(x-x_{j+1})}{(x_{j+1/2}-x_j)(x_{j+1/2}-x_{j+1})}$$

$$+ y_{j+1} \frac{(x-x_j)(x-x_{j+1/2})}{(x_{j+1}-x_j)(x_{j+1}-x_{j+1/2})} \qquad (6.18)$$

Une telle approximation de y conduit à une loi de convergence en $O(h^4)$.

6.2.3.2 Fonction de Lagrange d'ordre 3

L'utilisation de fonctions de base d'ordre 3 (voir figure 6.6) conduit à une loi de convergence en $O(h^6)$. La fonction y dans l'intervalle $x_j \leq x \leq x_{j+1}$ est alors représentée par

$$y = y_j\, g_j + y_{j+1/3}\, g_{j+1/3} + y_{j+2/3}\, g_{j+2/3} + y_{j+1}\, g_{j+1} \qquad (6.19)$$

ou

$$y = y_j \frac{(x-x_{j+1/3})(x-x_{j+2/3})(x-x_{j+1})}{(x_j-x_{j+1/3})(x_j-x_{j+2/3})(x_j-x_{j+1})} + y_{j+1/3} \frac{(x-x_j)(x-x_{j+2/3})(x-x_{j+1})}{(x_{j+1/3}-x_j)(x_{j+1/3}-x_{j+2/3})(x_{j+1/3}-x_{j+1})}$$

$$+ y_{j+2/3} \frac{(x-x_j)(x-x_{j+1/3})(x-x_{j+1})}{(x_{j+2/3}-x_j)(x_{j+2/3}-x_{j+1/3})(x_{j+2/3}-x_{j+1})} + y_{j+1} \frac{(x-x_j)(x-x_{j+1/3})(x-x_{j+2/3})}{(x_{j+1}-x_j)(x_{j+1}-x_{j+1/3})(x_{j+1}-x_{j+2/3})}. \qquad (6.20)$$

Nous connaissons maintenant la règle qui permet de produire des éléments de Lagrange d'ordre 4 ou plus.

Les éléments de Lagrange sont simple à créer. Ils conduisent à des solutions continues avec des dérivées premières discontinues. Par intervalle il faut en moyenne p inconnues si p est l'ordre du polynôme choisi (p = 1 pour les éléments linéaires, p = 2 pour les éléments paraboliques, etc.). Ces éléments sont applicables s'il y a au plus des dérivées premières dans la forme faible.

6.2.4 ELEMENTS DE HERMITE

Pour des problèmes avec des dérivées secondes dans la forme faible, la solution y doit être dérivable deux fois. Les éléments de Lagrange ne le sont pas. Il faut dans ce cas-ci choisir des éléments de Hermite. On demande que non seulement la fonction soit continue partout mais aussi sa dérivée. Sur chaque point j du réseau on place deux inconnues: la fonction y_j et sa dérivée y^*_j. Les fonctions de base sont choisies telles que la valeur soit 1 et la dérivée nulle ou la valeur soit 0 et la dérivée 1 (voir figure 6.7). L'inconnue y s'écrit dans $x_j \leq x \leq x_{j+1}$

$$y = y_j h_j + y_{j+1} h_{j+1} + y^*_j h^*_j + y^*_{j+1} h^*_{j+1} \qquad (6.21)$$

ou

$$y(x) = y_j \frac{(x-x_{j+1})^2 (x_{j+1}-3x_j+2x)}{(x_{j+1}-x_j)^3} + y_{j+1} \frac{(x-x_j)^2 (3x_{j+1}-x_j-2x)}{(x_{j+1}-x_j)^3}$$

$$+ y^*_j \frac{(x-x_j)(x-x_{j+1})^2}{(x_{j+1}-x_j)^2} + y^*_{j+1} \frac{(x-x_{j+1})(x-x_j)^2}{(x_{j+1}-x_j)^2} \qquad (6.22)$$

La représentation (6.22) est un polynôme cubique dont la fonction et sa dérivée sont continues partout. La loi de convergence est en $O(h^6)$. En moyenne, il faut deux inconnues par intervalle, une de moins que pour les éléments de Lagrange d'ordre 3 qui ont la même loi de convergence.

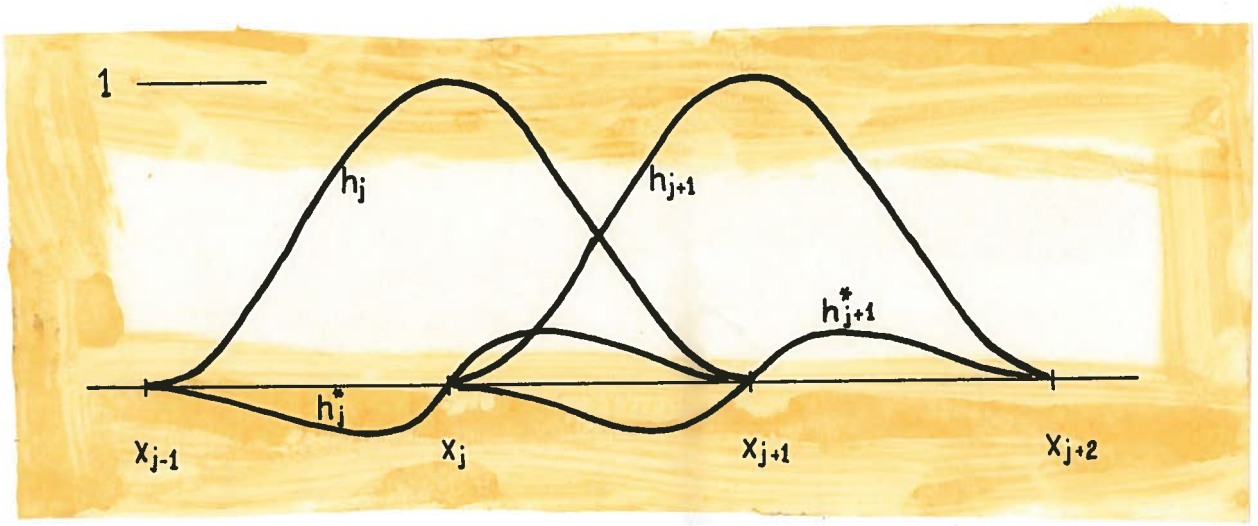


Fig. 6.7 : Les quatres éléments de Hermite non nuls dans l'intervalle $x_j \leq x \leq x_{j+1}$. Tous les éléments s'entendent sur deux intervalles. Les éléments h_j et h_{j+1} valent 1 aux noeuds correspondants avec une dérivée 0. Les éléments h_j^* et $h_j^*{}_{+1}$ sont nuls sur tous les noeuds et ont une dérivée 1 aux noeuds correspondants et une dérivée zéro aux autres noeuds.

6.2.5 ACCUMULATION DES POINTS DU RESEAU

La méthode des éléments finis offre le grand avantage suivant: l'ordre de convergence de la solution ($O(h^2)$ pour des éléments linéaires) est indépendant de l'équidistance du réseau.

Il est ainsi possible de concentrer les points du réseau autour de l'endroit où la fonction varie fortement. Néanmoins, il faut se montrer prudent quant au choix de cette accumulation. Une densification trop abrupte du réseau peut provoquer l'introduction de grandes erreurs sur la solution ou même l'apparition d'une solution parasite aux jonctions des différentes zones de densité.

Cet effet peut être évité par le choix d'une densification continue en introduisant une fonction de densité du réseau D(x), positive et normalisée à un:

$$\int_{x_o}^{x_N} D(x)\,dx = 1 \qquad (6.23)$$

En subdivisant en N intervalles, on obtient la valeur x_{j+1} à partir de x_j par

$$\int_{x_j}^{x_{j+1}} D(x)\,dx = 1/N \qquad (6.24)$$

Le procédé est illustré sur la figure 6.8. En partant de $x_o = 0$, on intègre D(x) jusqu'à ce que l'intégrale vaille 1/N. C'est pour cette valeur de x que l'on place le point x_1. On répète ce procédé: lorsque l'intégrale de D(x) vaut j/N, on y place le point x_j.

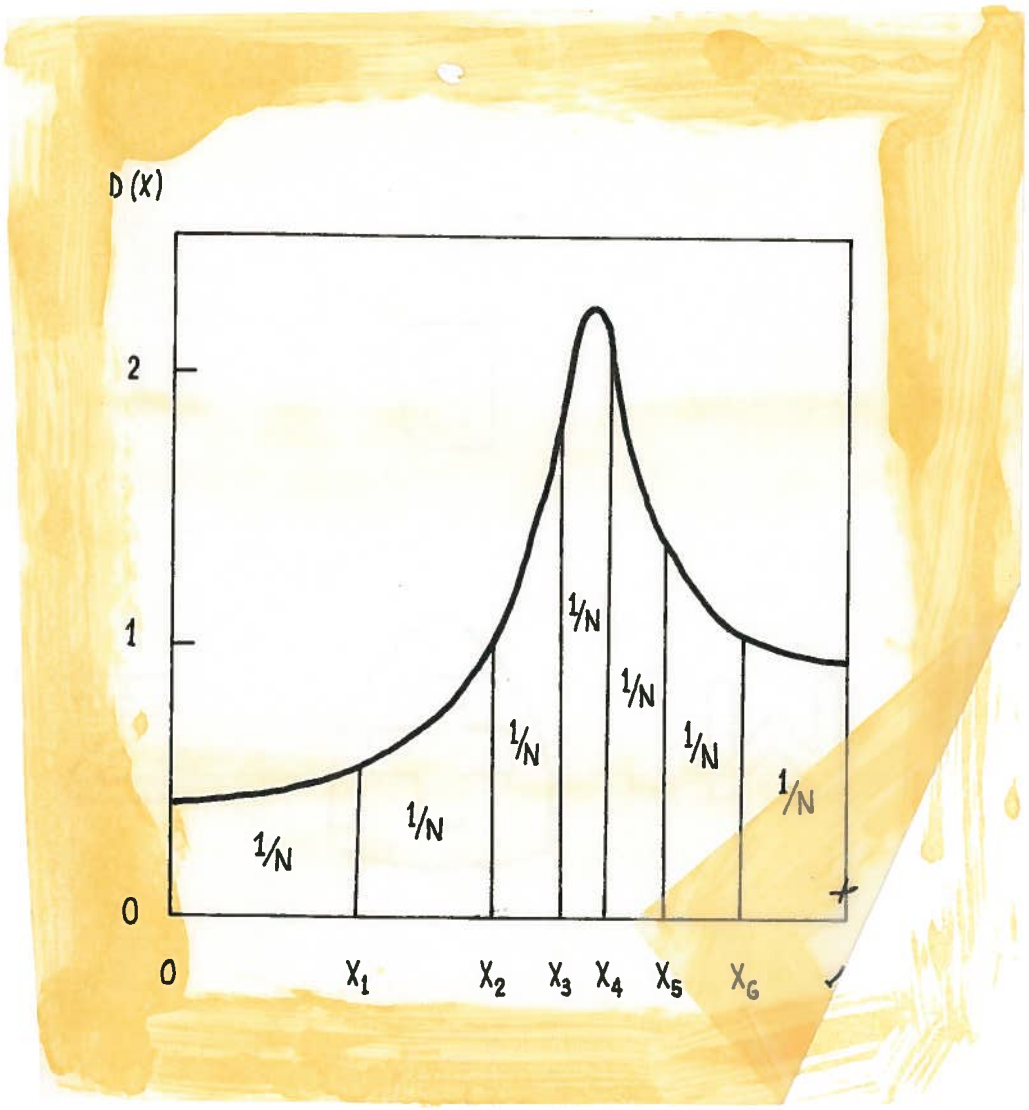


Fig. 6.8: Accumulation des points du réseau par l'aide d'une fonction de densité D(x).

6.3 ELEMENTS FINIS BIDIMENSIONNELS (Domaine rectangulaire)

6.3.1 FORME FAIBLE DU PROBLEME DE L'EQUILIBRE MHD

Au chapitre 3, nous avons reconstitué numériquement la solution analytique de Solovev (eq. 3.6) du problème de l'équilibre MHD (eq. 3.5) en appliquant la méthode des différences finies. Par une telle discrétisation, l'opérateur, qui est a priori symétrique, a été rendu non-symétrique. Cela peut être évité en appliquant la méthode des éléments finis.

Soit Ω le domaine rectangulaire (voir figure 3.9) dans lequel l'opérateur (voir eq. 3.5)

$$L(\psi) = \frac{1}{r}\frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial \psi}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2 \psi}{\partial z^2} = -\frac{dp}{d\psi} - \frac{1}{r^2} T \frac{dT}{d\psi} \qquad (6.25)$$

est défini. Comme condition sur le bord $\delta\Omega$, on prescrit ψ.

Soit $L^2(\Omega)$ l'ensemble de toutes les fonctions qui sont carrées intégrables sur Ω et V l'ensemble de toutes les fonctions $f \in L^2(\Omega)$, $\partial f/\partial r \in L^2(\Omega)$, $\partial f/\partial z \in L^2(\Omega)$ et f donnée sur $\delta\Omega$.

Le problème (6.25) peut alors être reformulé:

"Chercher $\psi(r,z) \in V$ tel que

$$\iint_{\Omega}\left[\left(\frac{1}{r}\frac{\partial \eta}{\partial r}\right)\left(\frac{1}{r}\frac{\partial \psi}{\partial r}\right) + \left(\frac{1}{r}\frac{\partial \eta}{\partial z}\right)\left(\frac{1}{r}\frac{\partial \psi}{\partial z}\right) - \eta\left(\frac{dp}{d\psi} + \frac{1}{r^2} T \frac{dT}{d\psi}\right)\right] rdrdz = 0, \qquad (6.26)$$

pour toute fonction $\eta \in V$ avec $\eta = 0$ sur $\delta\Omega$".

Cette formulation s'appelle formulation faible ou variationnelle ou encore de Galerkin.

Pour démontrer que (6.26) équivaut à (6.25), les deux premiers termes de (6.26) sont intégrés par partie. Le problème (6.26) s'écrit alors:

$$\int\int_{\Omega}\eta\Big[\frac{1}{r}\frac{\partial}{\partial r}\Big(\frac{1}{r}\frac{\partial\psi}{\partial r}\Big)+\frac{1}{r^2}\frac{\partial^2\psi}{\partial z^2}+\frac{dp}{d\psi}+\frac{1}{r^2}\,T\,\frac{dT}{d\psi}\Big]\,rdrdz$$

$$+\Big[\frac{\eta}{r}\Big(\frac{\partial\psi}{\partial r}+\frac{\partial\psi}{\partial z}\Big)\Big]_{\delta\Omega}=0, \qquad (6.27)$$

pour tout $\eta \in L^2$ et $\eta = 0$ sur $\delta\Omega$.

Puisque $\eta = 0$ sur $\delta\Omega$, le terme intégré tombe. La forme dite forte doit s'annuler pour tout η, donc l'intégrand doit s'annuler sur Ω, ce qui nous ramène au problème initial (6.25).

Pour introduire simplement les conditions aux limites, nous effectuons une transformation de variable

$$\tilde{\psi}(r,z) = \psi(r,z) - g(r,z) \qquad (6.28)$$

où $g(r,z)$ est une fonction dans V quelconque qui satisfait aux mêmes conditions du bord que ψ. Les conditions du bord sur $\tilde{\psi}(r,z)$ deviennent simplement

$$\tilde{\psi}(\delta\Omega) = 0. \qquad (6.29)$$

Avec la transformation (6.28) la formulation variationnelle (6.26) devient:

"Chercher $\tilde{\psi}(r,z) \in V$ avec $\tilde{\psi} = 0$ sur $\delta\Omega$ tel que

$$\int\int_{\Omega}\Big[\Big(\frac{1}{r}\frac{\partial\eta}{\partial r}\Big)\Big(\frac{1}{r}\frac{\partial\psi}{\partial r}\Big)+\Big(\frac{1}{r}\frac{\partial\eta}{\partial z}\Big)\Big(\frac{1}{r}\frac{\partial\psi}{\partial z}\Big)+\Big(\frac{1}{r}\frac{\partial\eta}{\partial r}\Big)\Big(\frac{1}{r}\frac{\partial g}{\partial r}\Big)+\Big(\frac{1}{r}\frac{\partial\eta}{\partial z}\Big)\Big(\frac{1}{r}\frac{\partial g}{\partial z}\Big)$$

$$-\eta\Big(\frac{dp}{d\psi}+\frac{1}{r^2}\,T\,\frac{dT}{d\psi}\Big)\Big]\,rdrdz = 0 \qquad (6.30)$$

pour tout $\eta \in V$ avec $\eta = 0$ sur $\delta\Omega$".

En passant de ψ à $\tilde{\psi}$, nous avons ajouté les deux termes contenant g au membres de droite.

6.3.2 ELEMENTS FINIS LINEAIRE SUR DOMAINE RECTANGULAIRE

Pour faciliter l'introduction des éléments finis bi-dimensionnels, nous considérons d'abord un domaine Ω rectangulaire $r_{min} \leq r \leq r_{max}$ et $-z_{max} \leq z \leq z_{max}$ (voir figure 6.9). Sur le bord on prescrit

$$\psi(r_{min}, z) = \frac{\psi_s}{a^2}\left[\frac{r_{min}^2}{E^2} z^2 + (r_{min}^2 - 1)^2/4\right]$$

$$\psi(r_{max}, z) = \frac{\psi_s}{a^2}\left[\frac{r_{max}^2}{E^2} z^2 + (r_{max}^2 - 1)^2/4\right] \qquad (6.31)$$

$$\psi(r, -z_{max}) = \psi(r, z_{max}) = \frac{\psi_s}{a^2}\left[\frac{z_{max}^2}{E^2} r^2 + (r^2 - 1)^2/4\right].$$

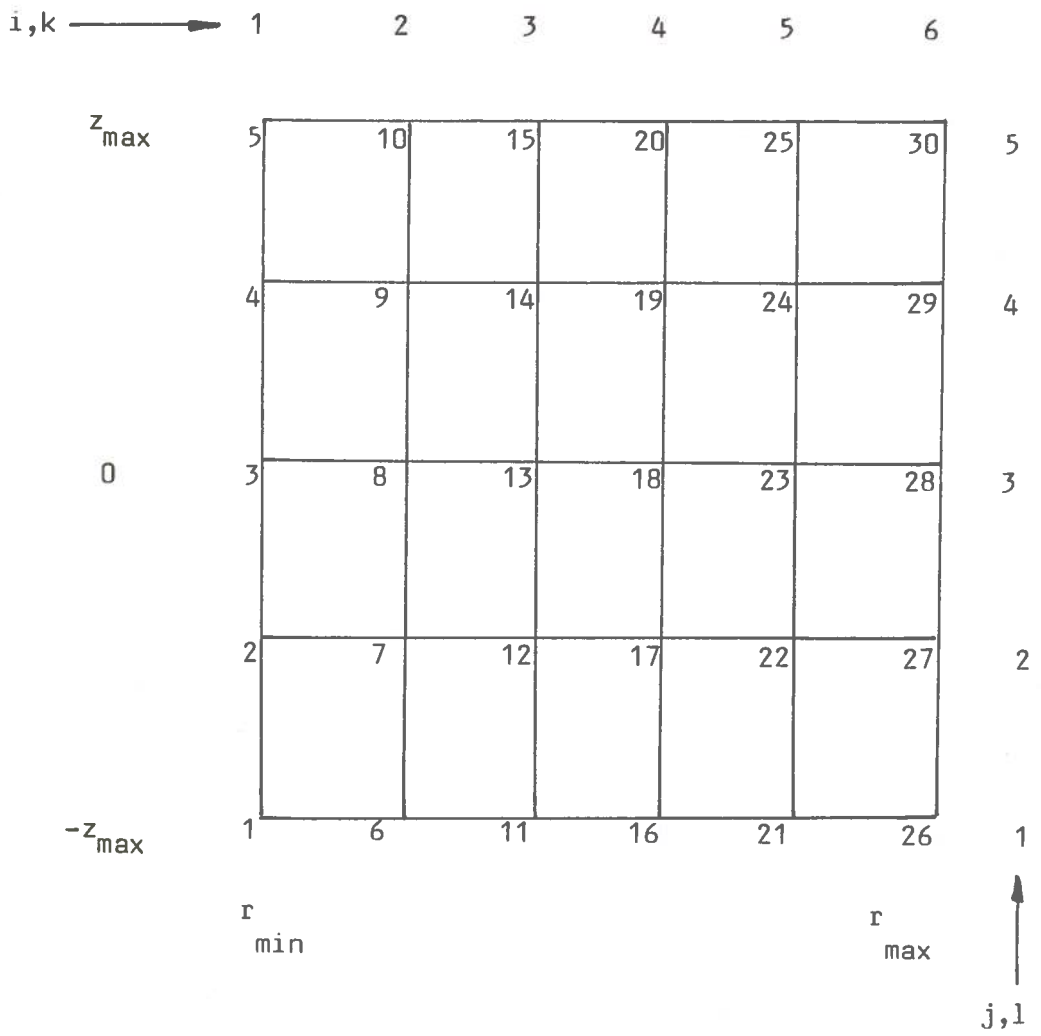
Ce domaine a déjà été choisi dans chapitre 3.5.5 lorsqu'on a discrétisé le problème de l'équilibre MHD par différences finies et résolu le système $\underline{A}\underline{x} = \underline{b}$ par la méthode MULTIGRID (voir aussi Appendice 3.C).

Subdivisions ce domaine en un maillage équidistant en r et z. En choisissant N_r intervalles en r et N_z intervalles en z, les coordonnées (r_i, z_j) du réseau sont

$$r_i = r_{min} + (i - 1)\,\frac{r_{max} - r_{min}}{N_r}, \quad 1 \leq i \leq N_r + 1$$

$$z_j = -z_{max} + (j - 1)\,\frac{2\, z_{max}}{N_z}, \quad 1 \leq j \leq N_z + 1. \qquad (6.32)$$

Sur chaque point de ce réseau on place une inconnue ψ_{ij}. Les conditions du bord s'écrivent alors



Domaine rectangulaire

Fig. 6.9 :

Numérotations du réseau

$$\left.\begin{aligned} \psi(r_{min},z) &\equiv g(r_{min},z) = \psi_{1j} \\ \psi(r_{max},z) &\equiv g(r_{max},z) = \psi_{N_r+1j} \end{aligned}\right\} \quad 1 \leq j \leq N_z + 1$$

$$\left.\begin{aligned} \psi(r,-z_{max}) &\equiv g(r,-z_{max}) = \psi_{i1} \\ \psi(r,z_{max}) &\equiv g(r,z_{max}) = \psi_{iN_z+1} \end{aligned}\right\} \quad 1 \leq i \leq N_r + 1 \qquad (6.33)$$

Nous pouvons développer la fonction $\tilde{\psi}$ inconnue en une combinaison linéaire de fonctions de base bilinéaires

$$\tilde{\psi}(r,z) = \sum_{i=2}^{N_r} \sum_{j=2}^{N_z} \tilde{\psi}_{ij}\, e_{ij}(r,z). \qquad (6.34)$$

Par le choix d'une discrétisation rectangulaire en r et z, les éléments de base $e_{ij}(r,z)$ peuvent s'écrire comme un produit d'éléments uni-dimensionnels, donc

$$e_{ij}(r,z) = e_i(r) * e_j(z), \qquad (6.35)$$

dont $e_i(r)$ et $e_j(z)$ sont définis par les équations (6.8) et représentés à la figure 6.3.

La figure 6.10 montre de manière schématique la dépendance fonctionnelle de l'élément de base $e_{ij}(r,z)$. Toute coupure r = constant ou z = constant donne deux lignes droites. Une coupure diagonale par contre conduit à une intersection parabolique.

Pour chaque fonction de test $\eta(r,z)$ choisie pour résoudre (6.30), on obtient une équation linéaire. Afin de déterminer les valeurs de toutes les inconnues, on doit choisir autant de fonctions de test qu'il y a d'inconnues, donc $(N_r-1) * (N_z-1)$ dans notre cas. Comme fonctions de test, nous choisissons

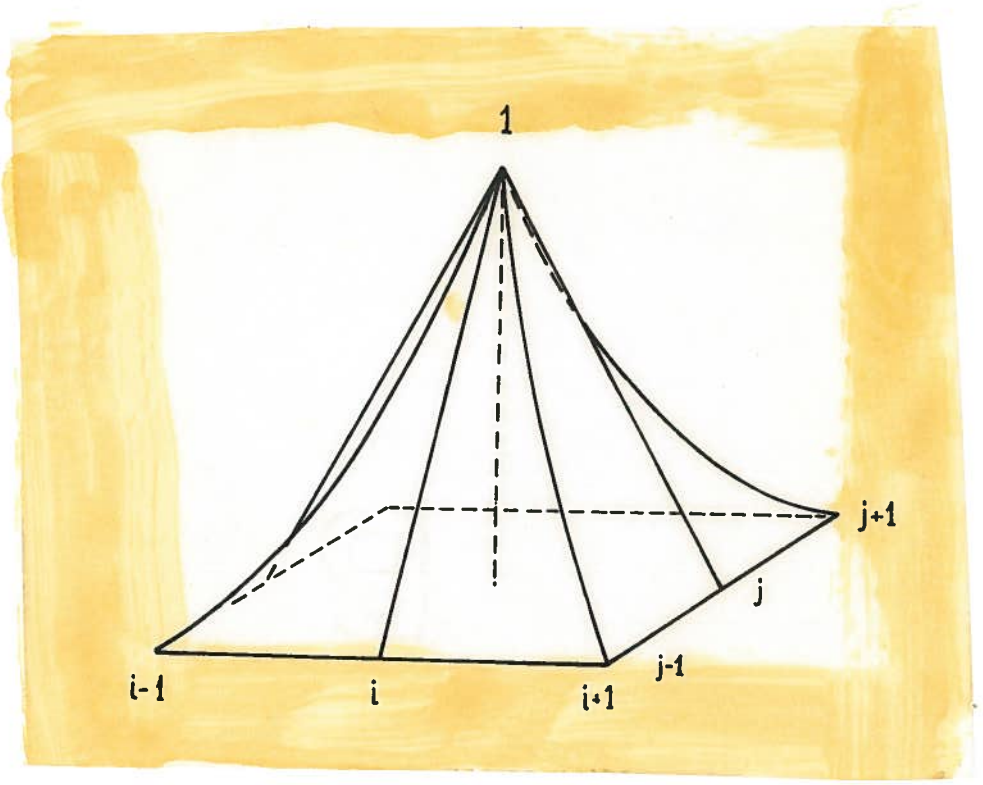


Figure 6.10 :

L'éléments de base bilinéaire $e_{ij}(r,z) = e_i(r) * e_j(z)$.

$$\eta_{kl}(r,z) = e_{kl}(r,z),\ 2 \leq k \leq N_r,\ 2 \leq l \leq N_z \qquad (6.36)$$

Un tel choix de η_{kl} offre l'avantage de rendre symétrique la matrice du système d'équations linéaires Ax = b résultant.

Nous devons encore chosir la dépendance fonctionnelle de g(r,z). Nous proposons

$$g(r,z) = \sum_{i=1}^{N_r+1} \sum_{j=1}^{N_z+1} g_{ij}\, e_{ij}(r,z) \qquad (6.37)$$

avec

$$g_{ij} = 0, \qquad 2 \leq i \leq N_r, \qquad 2 \leq j \leq N_z.$$

Cela veut dire que g(r,z) prend les valeurs du bord (6.33) et décroît linéairement à zéro vers les points les plus proches du bord.

Le problème approché de (6.30) s'écrit maintenant

"Chercher $\tilde{\psi}_{ij}$ tel que

$$\sum_{i=2}^{N_r} \sum_{j=2}^{N_z} \tilde{\psi}_{ij}\, a_{ijkl} = - \sum_{i=1}^{N_r+1} \sum_{j=1}^{N_z+1} g_{ij}\, a_{ijkl} + b_{kl}$$

$$a_{ijkl} = \int_{r_{k-1}}^{r_{k+1}} \frac{1}{r} \frac{de_k(r)}{dr} \frac{de_i(r)}{dr}\, dr \int_{z_{l-1}}^{z_{l+1}} e_l(z)\, e_j(z)\, dz$$

$$+ \int_{r_{k-1}}^{r_{k+1}} \frac{1}{r} e_k(r)\, e_i(r)\, dr \int_{z_{l-1}}^{z_{l+1}} \frac{de_l(z)}{dz} \frac{de_j(z)}{dz}\, dz \qquad (6.38)$$

$$b_{kl} = \int_{r_{k-1}}^{r_{k+1}} r e_k(r)\, dr \int_{z_{l-1}}^{z_{l+1}} e_l(z) \left[\frac{dp}{d\psi} + \frac{1}{r^2} T \frac{dT}{d\psi} \right] dz$$

où k varie de 2 à N_r et l de 2 à N_z".

Notons que les g_{ij} sont nuls aux points où les ψ_{ij} sont non nuls et vice versa, donc

$$\begin{aligned} g_{ij} &= \psi_{ij} \quad \text{au bord} \\ \tilde{\psi}_{ij} &= \psi_{ij} \quad \text{à l'intérieur.} \end{aligned} \tag{6.39}$$

La formulation (6.38) conduit à un système de $(N_r-1) * (N_z-1)$, équations avec autant d'inconnues $\tilde{\psi}_{ij}$.

6.3.3 CONSIDERATIONS PRATIQUES

Comment résoudre (6.38) en pratique. Nous proposons le procédé suivant: revenons d'abord sur l'inconnue initiale $\psi = \tilde{\psi} + g$. Puis, introduisons des fonctions de test η_b fictives sur le bord. Nous considérons donc d'abord toutes les valeurs de ψ (le bord inclu) comme des inconnues. La formulation (6.38) se récrit:

"Chercher ψ_{ij} tel que

$$\sum_{i=1}^{N_r+1} \sum_{j=1}^{N_z+1} \psi_{ij}\, a_{ijkl} = b_{kl}$$

pour $1 \leq k \leq N_r + 1$, $1 \leq l \leq N_z + 1$ et $\psi_{ij} = g_{ij}$ sur $\delta\Omega$." (6.40)

Introduisons maintenant une numérotation continue du réseau tel que (voir figure 6.9)

$$\psi_{ij} \equiv \psi_m, \qquad m = (i-1)\,(N_z+1) + j. \tag{6.41}$$

Le système d'équations (6.40) devient

$$\sum_{m=1}^{(N_r+1)(N_z+1)} a_{m\,n}\, \psi_m = b_n, \qquad n = 1, \ldots, (N_r+1)\,(N_z+1), \tag{6.42}$$

avec

$$\psi_m = g_m \text{ sur } \delta\Omega .$$

Sous forme matricielle (6.36) s'écrit

$$A\underline{x} = \underline{b} \tag{6.43}$$

avec

$$\underline{x} = \underline{g} \text{ sur } \delta\Omega . \tag{6.44}$$

Avec la nouvelle variable

$$\underline{x} = \underline{x} - \underline{x}_b , \tag{6.45}$$

le système d'équations linéaires (6.43) devient

$$A\underline{x} = \underline{b} - A\underline{x}_b \tag{6.46}$$

avec

$$\underline{x}_b = 0 \text{ sur } \delta\Omega . \tag{6.47}$$

La condition du bord s'introduit en biffant les lignes et les colonnes de A et les composantes de $\underline{b}$ correspondant aux variables du bord. Afin de ne pas être obligé de comprimer A et $\underline{b}$, on place des uns sur la diagonale de A et la valeur du bord au bon endroit dans $\underline{b}$. Ceci revient à imposer les valeurs du bord en tant que solution du système. La solution obtenue contient alors toute la solution (point de bord compris) ψ_m, m = 1,..., (N_r+1) (N_z+1). Il faut encore noter qu'en imposant $\underline{x}_b = 0$, on a en fait éliminé la contribution de la fonction de test η_b fictive du système.

L'intégration numérique des doubles intégrales (6.38) se fait par une formule de Gauss à quatre points (voir chapitre 2).

6.4 LES ELEMENTS ISOPARAMETRIQUES (Domaine quelconque)

6.4.1 LE DOMAINE COURBE ET SA DISCRETISATION

Considérons maintenant le domaine Ω courbe (Fig. 6.11). Le bord $\delta\Omega$ de Ω est une surface de flux $\psi = \psi_S$ = constant ou $\psi=0$ si $\psi = \psi - \psi_S$. Discrétisons ce domaine en choisissant un point R_c sur l'axe $z = 0$ à l'intérieur de Ω. Partant de R_c, on trace des rayons à certains angles θ_j, $j = 1, \ldots, N_\theta$ jusqu'au bord qui sont subdivisés en N_p intervalles. La discrétisation résultante est illustrée sur la Figure 6.11. Chaque cellule du réseau est un quadrangle quelconque avec quatre coins ayant les coordonnées (r_1, z_1), (r_2, z_2), (r_3, z_3), (r_4, z_4) (voir Fig. 6.12).

Notons que la surface du domaine diminue en l'approximant par des bouts droits. En particulier, la surface d'un cercle devient $\rho_s^2 \left[N_\theta \sin(2\pi/N_\theta)\right]/2$ au lieu de $\pi\rho_s^2$ si l'on effectue une discrétisation équidistante en θ.

6.4.2 ELEMENTS ISOPARAMETRIQUES

Afin de pouvoir effectuer les intégrations simplement une fois pour toute, nous projetons un quadrangle sur un carré unitaire $0 \leq \xi \leq 1$, $0 \leq \chi \leq 1$ (Fig. 6.12). L'ansatz bilinéaire s'écrit

$$\begin{aligned} r(\xi,\chi) &= \alpha_1 + \alpha_2\xi + \alpha_3\chi + \alpha_4\xi\chi \\ z(\xi,\chi) &= \beta_1 + \beta_2\xi + \beta_3\chi + \beta_4\xi\chi . \end{aligned} \tag{6.48}$$

Les huit paramètres α_1 à α_4 et β_1 à β_4 sont déterminés par les valeurs des coordonnées r_1 à r_4 et z_1 à z_4 :

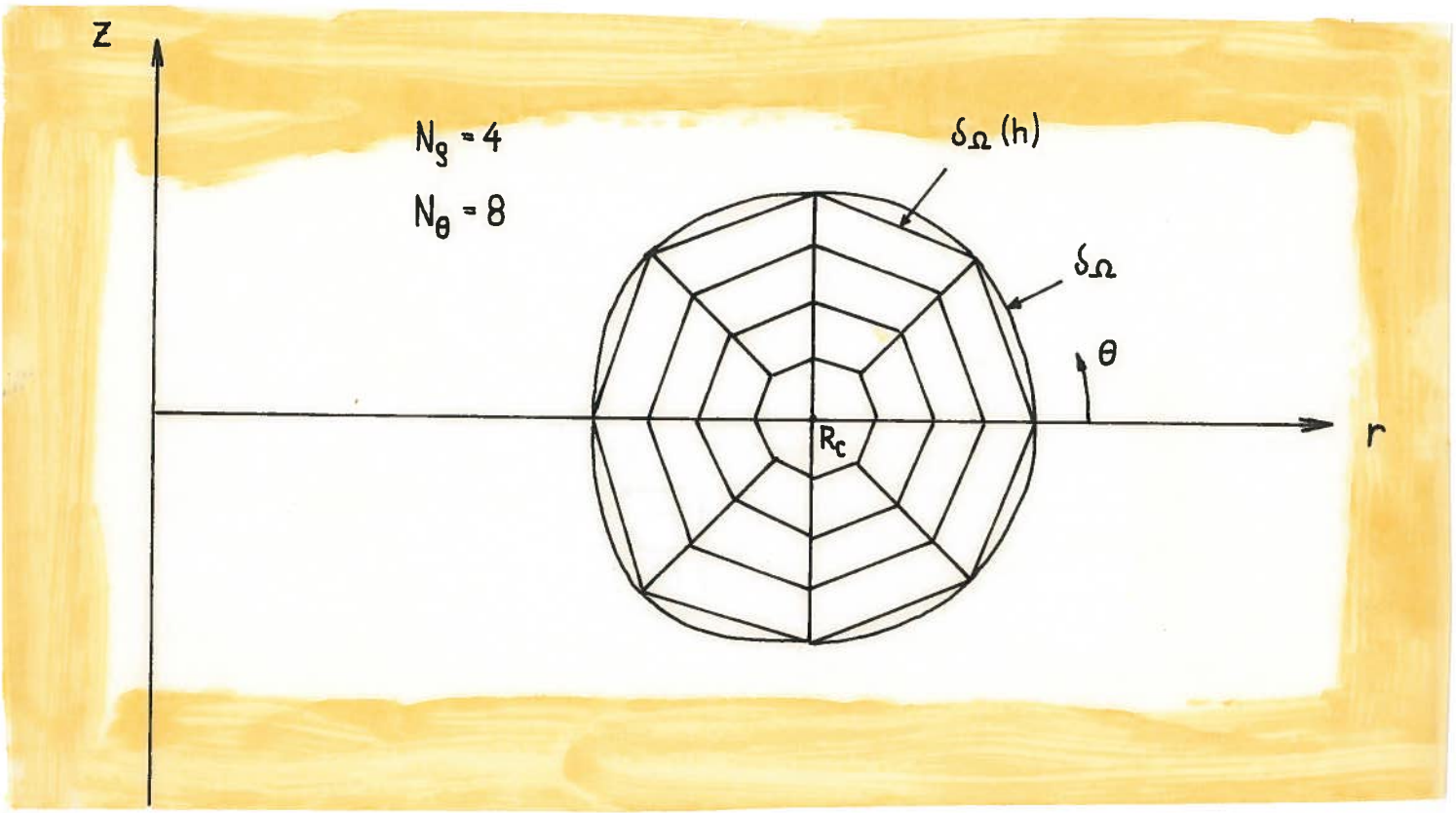


Fig. 6.11 : Domaine courbe: Discrétisation dans le plan (r,z)

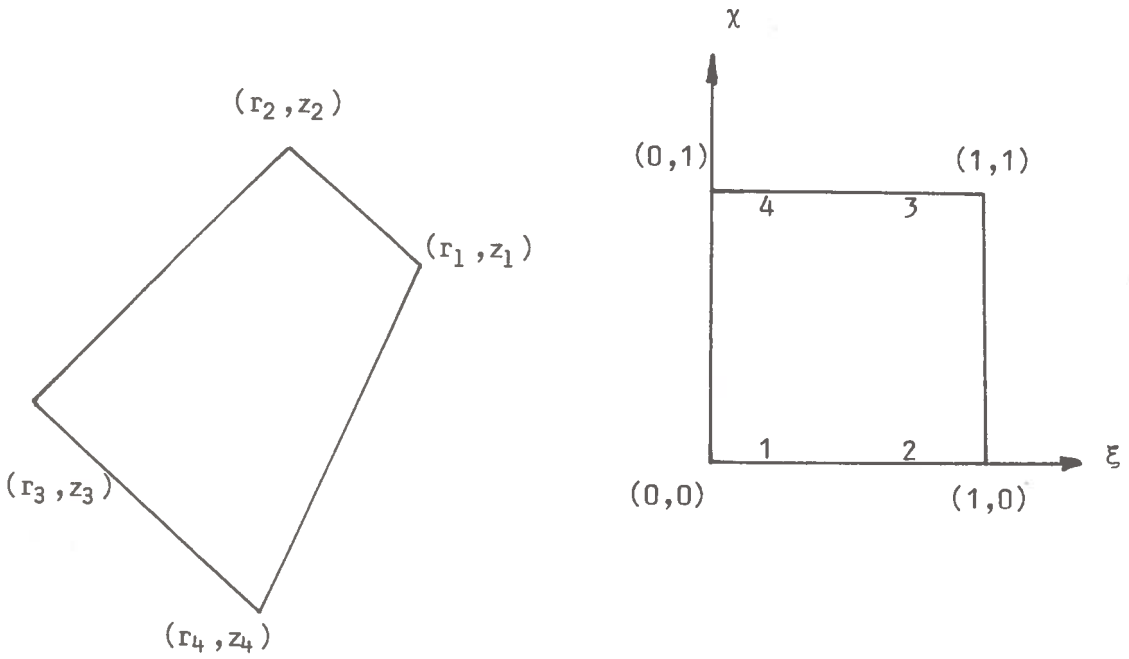


Fig. 6.12 : Les éléments isoparamétriques: Projection du quadrangle en r-z sur ξ-χ.

$$
\begin{aligned}
\alpha 1 &= r_1 \\
\alpha_2 &= r_2 - r_1 \\
\alpha_3 &= r_3 + r_1 - r_2 - r_4 \\
\alpha_4 &= r_4 - r_1
\end{aligned}
\qquad (6.49)
$$

$$
\begin{aligned}
\beta_1 &= z_1 \\
\beta_2 &= z_2 - z_1 \\
\beta_3 &= z_3 + z_1 - z_2 - z_4 \\
\beta_4 &= z_4 - z_1
\end{aligned}
$$

Pour éviter des problèmes d'intégration et d'unicité de (6.48), nous demandons que tous les angles des quadrangles soient plus petits que π.

Lors de la transformation de (r,z) en (ξ,χ), l'élément de surface devient

$$drdz = Jd\,\xi\,d\,\chi \qquad (6.50)$$

où

$$J = \frac{1}{\frac{\partial \xi}{\partial r}\frac{\partial \chi}{\partial z} - \frac{\partial \xi}{\partial z}\frac{\partial \chi}{\partial r}} \qquad (6.51)$$

est le Jacobien.

En pratique, on traite quadrangle après quadrangle. Du plan (r,z) on passe au plan (ξ,χ) où on effectue tous les intégrales des contributions de la matrice A et du membre de droite $\underline{b}$. Pour ce faire numériquement, on utilise une formule de Gauss à quatre points.

En appliquant cette méthode à notre problème de l'équilibre MHD, nous remarquons que l'erreur est de 10% si l'on discrétise le domaine en 60 quadrangles. Cette erreur importante est principalement due à la troncation de la surface. En prenant huit intervalles équidistants en θ, la surface d'un cercle diminue de ~ 10%.

6.4.3 TRANSFORMATION DE VARIABLES

Pour éviter la perte en surface due au choix du système de coordonnées (r,z) (voir Fig. 6.11), on pourrait directement utiliser le système de coordonnées polaires (ρ,θ) avec la correspondance

$$r = R_c + \rho \cos\theta$$
$$z = \rho \sin\theta \qquad (6.52)$$

Dans ce système, la formulation (6.26) s'écrit:

"Chercher $\psi(\rho,\theta) \in V$ tel que

$$\iint_\Omega \left[\left(\frac{1}{r}\frac{\partial\eta}{\partial\rho}\right)\left(\frac{1}{r}\frac{\partial\psi}{\partial\rho}\right) + \left(\frac{1}{r\rho}\frac{\partial\eta}{\partial\theta}\right)\left(\frac{1}{r\rho}\frac{\partial\psi}{\partial\theta}\right) - \eta\left(\frac{dp}{d\psi} + \frac{1}{r^2} T \frac{dT}{d\psi}\right)\right] r\rho\, d\rho\, d\theta = 0 \qquad (6.53)$$

pour toute $\eta \in V$ avec $\eta = 0$ sur $\delta\Omega$".

Le passage de (ρ,θ) à (ξ,χ) se fait de la même façon que sous (6.48) et (6.49). La discrétisation en quadrangles dans le plan (ρ,θ) se fait comme sur la Figure 6.13 pour un cas presque circulaire ($E = 1$, $a = 1/3$, $p_0 = -2$, paramètres définis dans eqs. 3.6 et 3.7). On voit que la perte en surface est minime. Comme conséquence, le résultat obtenu avec $N_\rho = N_\theta = 8$ intervalles n'a plus qu'une erreur de 1.3% ($\psi_s = .05486$ au lieu de 1/18 = .05556). Si l'on prend $N_\rho = 16$ et $N_\theta = 8$, l'erreur passe à 0.2% ($\psi_s = .05554$)!

L'idéal serait de choisir un système de coordonnées dans lequel la surface est une constante, par exemple (ψ,θ). Ce système de coordonnée est non-orthogonal. Comme solution on obtiendrait $\rho(\psi,\theta)$.

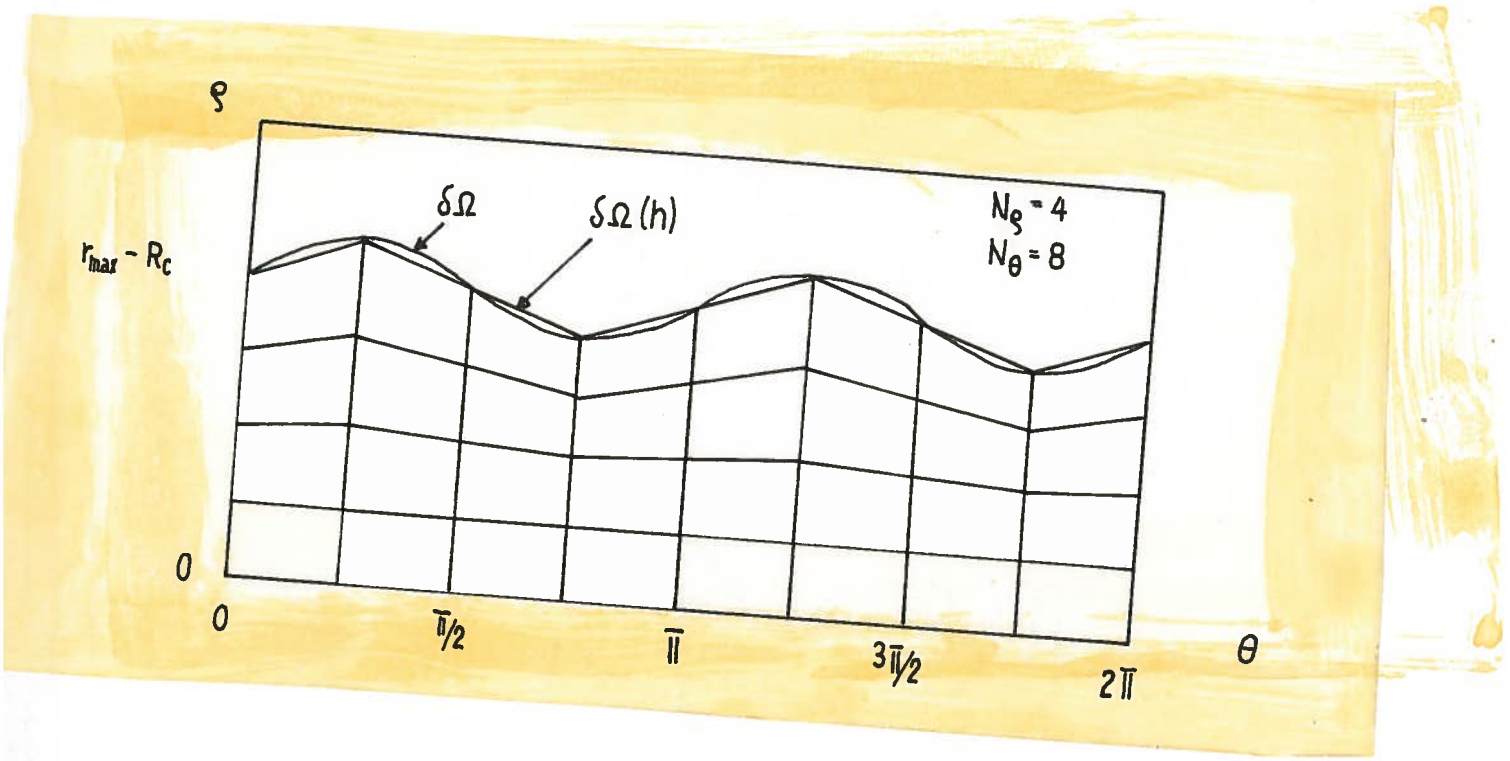


Fig. 6.13: Discrétisation du plan (ρ, θ) en quadrangles.
Le point $\rho = 0$ est un seul point dégénéré.

6.4.4 TECHNIQUES DE PROGRAMMATION

Un programme basé sur les éléments finis s'écrit d'une façon structurée et modulaire, afin de pouvoir aisément le modifier. Cela implique que, par exemple, les éléments, la méthode d'intégration, les termes de la forme variationnelle, les conditions aux limites ou encore la définition du réseau soient programmés dans différentes sous-routines.

Comme illustration, nous allons discuter l'organisation du code d'éléments finis résultant de l'équilibre MHD dans un domaine rectangulaire. Les intégrales à effectuer sont celles des équations (6.38). En pratique, on se place dans une cellule et on effectue toutes les intégrales qui s'ajoutent à la matrice A et au membre de droite $\underline{b}$ sans se soucier des conditions du bord. Celles-ci seront imposées plus tard à l'aide d'un module séparé.

Considérons la cellule $r_i \leq r \leq r_{i+1}$, $z_j \leq z \leq z_{j+1}$ (voir Fig. 6.14). Aux quatre noeuds se trouvent les inconnues ψ_{ij}, ψ_{i+1j}, ψ_{ij+1} et ψ_{i+1j+1} que l'on renumérote localement ψ_1, ψ_2, ψ_3 et ψ_4. Il y a quatre éléments de base $e_1 = e_{ij}$, $e_2 = e_{i+1j}$, $e_3 = e_{ij+1}$ et $e_4 = e_{i+1j+1}$ qui sont non nuls dans cette cellule. Avec les quatre fonctions de test η_1, η_2, η_3 et η_4 locales et non nulles, il y a donc 16 contributions à la matrice A provenant de cette cellule et quatre contributions au membre de droite.

Les 16 contributions à la matrice A sont calculées terme par terme donc en deux pas:

D'abord on calcule

$$W_1 = \int_{r_i}^{r_{i+1}} \int_{z_j}^{z_{j+1}} \left(\frac{1}{r}\frac{\partial \eta}{\partial r}\right)\left(\frac{1}{r}\frac{\partial \psi}{\partial r}\right) r\,dr\,dz \qquad (6.54)$$

$$= \iint I_1\, r\,dr\,dz$$

et puis

$$W_2 = \int_{r_i}^{r_{i+1}} \int_{z_j}^{z_{j+1}} \left(\frac{1}{r}\frac{\partial \eta}{\partial z}\right)\left(\frac{1}{r}\frac{\partial \psi}{\partial z}\right) r\,dr\,dz \qquad (6.55)$$

$$= \iint I_2\, r\,dr\,dz.$$

Considérons d'abord seulement un terme, disons (6.55). Cette intégrale est résolue numériquement par une formule de Gauss à quatre points (voir fig. 6.14)

$$W_1 = \frac{\Delta r \Delta z}{4} \sum_{k=1}^{4} r_k\, I_1(r_k, z_k) \qquad (6.56)$$

En introduisant la variation fonctionnelle de l'inconnue

$$\psi(r,z) = \psi_1 e_1 + \psi_2 e_2 + \psi_3 e_3 + \psi_4 e_4 \qquad (6.57)$$

et les différentes fonctions de base η_1 à η_4 dans W_1 on obtient

$$W_1 = \frac{\Delta r \Delta z}{4} \sum_{k=1}^{4} r_k\, \mathcal{A}\, \underline{\psi} \qquad (6.58)$$

où $\mathcal{A}$ est une matrice 4x4 avec les éléments

$$a_{ij}(r_k, z_k) = \left[\left(\frac{1}{r}\frac{\partial e_i}{\partial r}\right)\left(\frac{1}{r}\frac{\partial e_j}{\partial r}\right)\right](r_k, z_k), \qquad (6.59)$$

qui sont les valeurs des intégrands. Le vecteur $\underline{\psi}$ a les quatre composantes ψ_1, ψ_2, ψ_3 et ψ_4. Introduisons le vecteur

$$\underline{v}^T(r_k, z_k) = \left(\frac{1}{r}\frac{\partial e_1}{\partial r}, \frac{1}{r}\frac{\partial e_2}{\partial r}, \frac{1}{r}\frac{\partial e_3}{\partial r}, \frac{1}{r}\frac{\partial e_4}{\partial r}\right)(r_k, z_k), \qquad (6.60)$$

alors la matrice locale $\mathcal{A}$ s'obtient simplement par une multiplication diadique de $\underline{v}$ avec lui-même

$$\mathcal{A} = \underline{v} \cdot \underline{v}^T. \qquad (6.61)$$

Dans l'appendice 6.12 nous montrons la construction de la contribution d'une cellule $W = W_1 + W_2$ à la matrice finale A. La sous-routine INTEGA organise le calcul de W (matrice 4 x 4) qui s'appelle XAB dans le code. Après initialisation (mise à zéro par RESETR, NVAL = 4), le calcul des coordonnées des points nodaux par DEFCEL et les points pour l'intégration de Gauss par DEFINT on effectue l'intégration (somme sur NPOINT = 4 points de Gauss). Pour chaque point de Gauss on calcule la matrice locale des intégrands $\mathcal{A}$ par AMATRX et on ajoute les 16 contributions.

Comment organise-t-on le calcul de $\mathcal{A}$? Dans AMATRX ($\mathcal{A}$ s'appelle XT) XT est mis à zéro. Dans BASIS on définit les valeurs des éléments de base e_1 à e_4 et leurs dérivées $\partial e/\partial r$ et $\partial e/\partial z$ au point de Gauss. Pour les deux termes W_1 et W_2, on définit dans CONST la forme des termes quadratiques. C'est précisément dans cette sous-routine que nous préscrivons la physique. Dans VECT, le vecteur $\underline{v}$ défini dans (6.60) est calculé et dans DIADIC on effectue la multiplication diadique, eq. (6.61).

Si l'on passe du système de coordonnées (r,z) à (ρ,θ), la forme variationnelle (6.25) devient celle écrite sous (6.53). Puisque toute la physique est définie dans CONST, le passage de (r,z) à (ρ,θ) implique la modification de CONST seulement. A la figure 6.15, où l'on a représenté la nouvelle sous-routine CONST, on voit que peu de modifications suffisent à faire passer du système (r,z) à (ρ,θ).

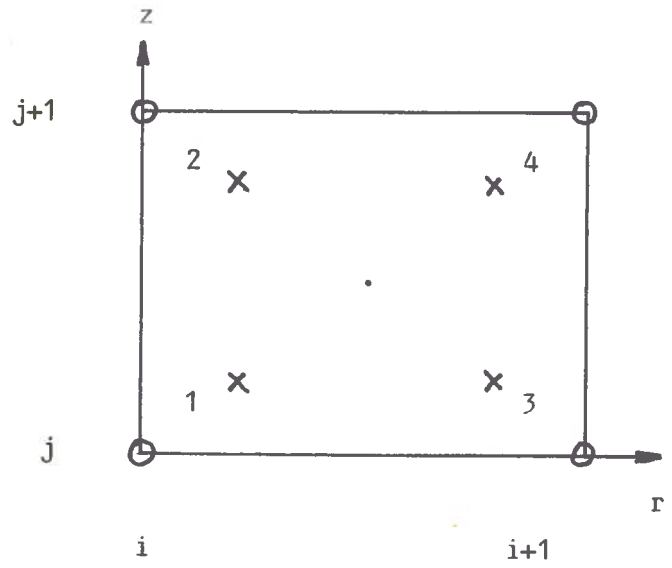


Fig. 6.14: Numérotation locale : Position des points nodaux ○

Points pour l'intégration de Gauss ×

```
      SUBROUTINE CONST(KTERM,PRHO,PT,PC)
C
C
C C2SB40  CONSTANT FOR QUADRATIC TERMS
C
C
*CALL COMBND
C
C    *    *    *    *    *    *    *    *    *
C
      DIMENSION PC(4)
C
      ZR=BPS(4)+PRHO*COS(PT)
      GO TO (100,200,300) KTERM
C
C     CONSTANT FOR FIRST SQUARE
C
  100 CONTINUE
      PC(1)=PRHO*ZR
      PC(2)=1.0/ZR
      RETURN
C
C     CONSTANT FOR SECOND SQUARE
C
  200 CONTINUE
      PC(1)=PRHO*ZR
      PC(3)=1.0/(PRHO*ZR)
      RETURN
C
C     CONSTANT FOR RIGHT HAND SIDE
C
  300 CONTINUE
      PC(4)=-PRHO*SOURCE(ZR)/ZR
      RETURN
C
      END
```

Fig. 6.15: Sous-routine CONST pour la formulation variationnelle en (ρ, θ).

6.5 PROBLEME NON LINEAIRE UNIDIMENSIONNEL: y" = Sh y

6.5.1 L'EXEMPLE : y" = Sh y

Considérons l'équation différentielle ordinaire

$$d^2y/dx^2 = y'' = \mathrm{Sh}\ y \tag{6.62}$$

où y(x) satisfait les conditions aux limites

$$\begin{aligned} y(0) &= 0 \\ y(10) &= 1 \ . \end{aligned} \tag{6.63}$$

C'est un problème non linéaire. Souvent on le nomme aussi quasi-linéaire puisque la nonlinéarité n'agit pas sur les plus hautes dérivées.

Des problèmes du type (6.62) et (6.63) peuvent, en général, être résolus par une méthode de tir, ce qui revient à remplacer la condition en x = 10 par une condition initiale

$$y'(0) = \varepsilon \tag{6.64}$$

en ajustant itérativement ε de telle façon que la condition y(10) = 1 soit satisfaite. Pouvons-nous appliquer cette méthode à notre exemple (6.62, 6.63)? La réponse est non.

Le problème (6.62) avec les conditions initiales y(0) = 0 et y'(0) = ε a la solution implicite

$$x = \int_0^y \frac{1}{\sqrt{4\ \mathrm{Sh}^2\ (u/2) + \varepsilon^2}}\ du \ . \tag{6.65}$$

La solution y est infinie pour (voir figure 6.16)

$$x(y = \infty) = \ell n(8/\varepsilon) \ . \tag{6.66}$$

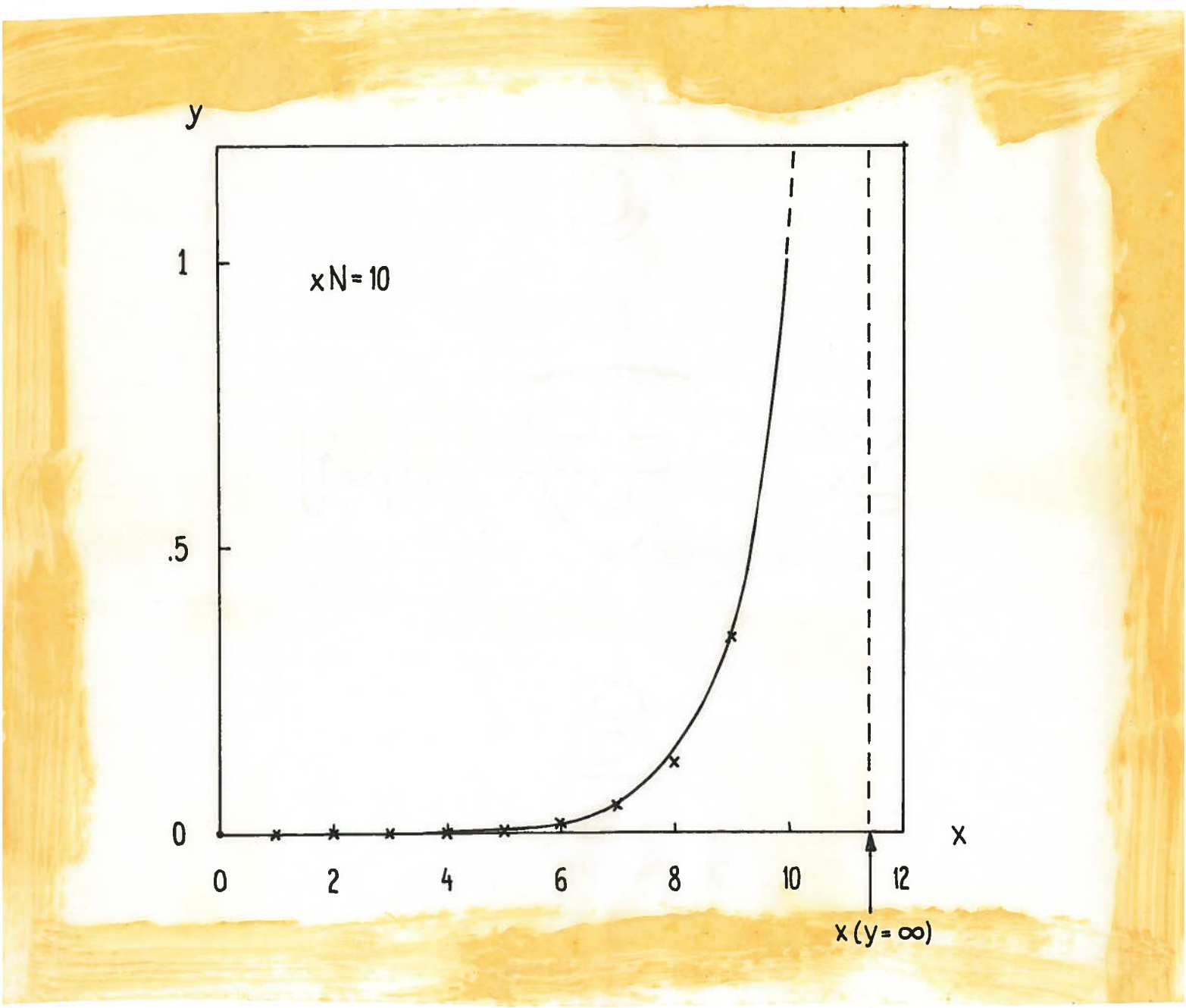


Fig. 6.16: Solution de $y'' = \mathrm{Sh}\, y$ avec $y(0) = 0$ et $y(10) = 1$.
Les croix représentent la solution numérique avec 10 intervalles équidistants.
$y = \infty$ à $x = 11.42$
$y'(0) = \varepsilon = 8.88 \times 10^{-5}$!

Cela veut dire que $0 < \varepsilon < 1/8\ e^{-10}$ pour satisfaire la condition $y(10) = 1$. Il s'agit là d'un problème typique dit de couche. De minimes erreurs numériques qui changent un peu la pente font démarrer la solution à l'infini bien avant $x = 10$. Il faut donc résoudre ce problème comme un problème non linéaire à condition aux limites. Pour traiter la nonlinéarité, il y a trois méthodes: la méthode de Picard, la méthode de Newton et la méthode de continuation.

6.5.2 METHODE DE PICARD

La méthode de Picard revient à utiliser le schéma itératif:

$$y''_{k+1} = Shy_k \tag{6.67}$$

à partir d'une solution initiale y_0. Une telle itération est analogue à la recherche du zéro d'une fonction

$$f(x) = 0 \tag{6.68}$$

itérativement par

$$x_{k+1} = g(x_k) = f(x_k) + x_k \tag{6.69}$$

en partant d'une solution approchée x_0. A la Figure 6.17 sont représentés les cas divergent, pour lequel

$$|g(x)| < |x|. \tag{6.70}$$

et convergent, pour lequel

$$|g(x)| < |x|. \tag{6.71}$$

Le problème $y'' = Shy$, $y(0) = 0$ et $y(10) = 1$ semble se trouver dans la situation stable puisque

$$|Shy| < |y|. \tag{6.72}$$

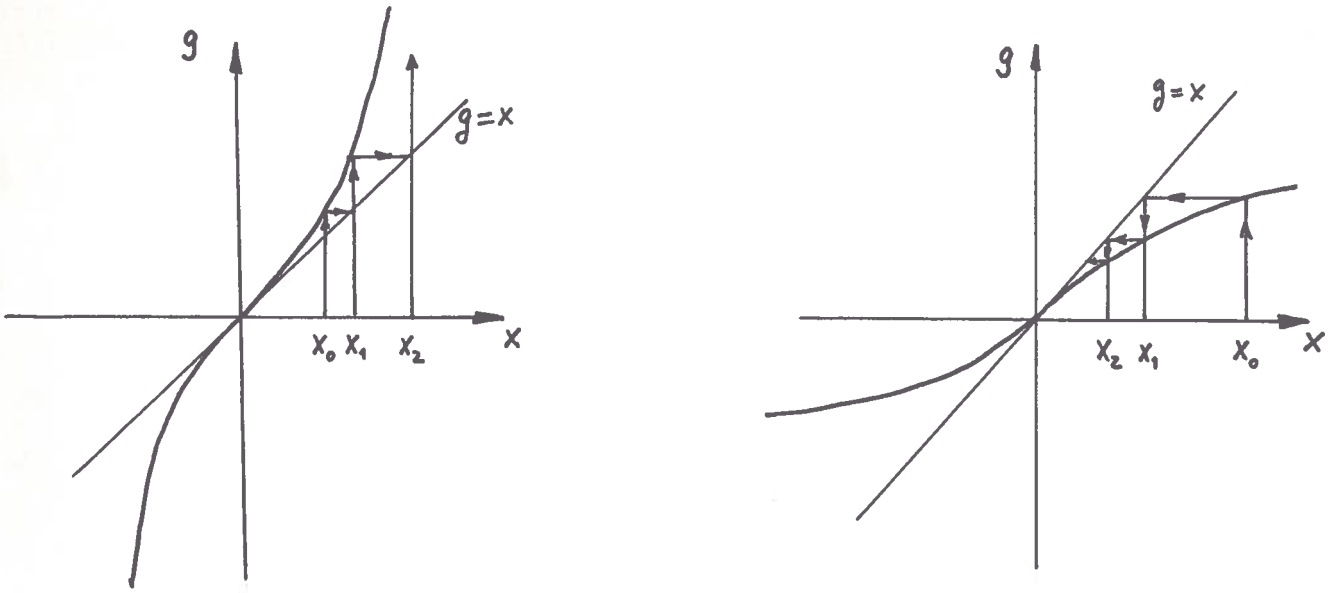


Fig. 6.17 : A gauche: Pas de convergence vers le point fixe.
A droite: Convergence vers le point fixe en x = 0.

Le schéma itératif (6.67) a été résolu par la méthode des éléments finis. En multipliant à gauche par la fonction test η en intégrant sur $x \in [0,10]$ et en intégrant par partie, on obtient la formulation variationnelle:

"Chercher $y_{k+1} \in V$, $y_{k+1}(0) = 0$, $y_{k+1}(10) = 1$, tel que

$$\int_0^{10} (y' y'_{k+1} + \eta \ \mathrm{Sh}\ y_k)\, dx = 0 \qquad (6.73)$$

pour tout $\eta \in V$ avec $\eta(0) = \eta(10) = 0$", dont V est l'espace de toutes les fonctions $f \in L^2(0,10)$ avec $df/dx \in L^2(0,10)$.

Cette formulation a été programmée pour un réseau equidistant dans $x \in (0,10)$ et une solution initiale

$$y_o = x/10 \tag{6.74}$$

La table 6.1 montre les premiers deux pas de l'itération. Le troisième pas donne une erreur d'overflow! Peut-être la solution initiale et le réseau sont insuffisants.

Essayons d'abord avec une accumulation de points. Comme fonction de densité D(x), (eq. 6.25), nous choisissons pour notre problème y" = Sh y:

$$D(x) = Ae^{\alpha x}. \tag{6.75}$$

La normalisation de D(x) entre $x_o = 0$ et $x_N = 10$ donne

$$A = \frac{\alpha}{e^{10\alpha} - 1} \tag{6.76}$$

L'intégrale (6.24) peut être effectué analytiquement et vaut

$$\int_{x_j}^{x_{j+1}} A\, e^{\alpha x}\, dx = 1/N = \frac{A}{\alpha}\left[e^{\alpha x_{j+1}} - e^{\alpha x_j}\right]. \tag{6.77}$$

Le nouveau point du réseau x_{j+1} s'obtient donc itérativement par

$$x_{j+1} = \ln\left[e^{\alpha x_j} + \frac{e^{10\alpha} - 1}{N}\right] / \alpha. \tag{6.78}$$

On peut se convaincre que pour $\alpha \to 0$ on obtient le réseau équidistant.

Table 6.1:

Itération de Picard avec $y_0 = x/10$, $\alpha = 0$, $\omega = 1$

x	y_0	y_1	y_2	y_3
0	.0	0	0	
1	.1	-1.6353	3796.34	
2	.2	-3.1704	749.72	
3	.3	-4.5040	1109.6	
4	.4	-5.5328	1421.5	overflow
5	.5	-6.1505	1608.6	
6	.6	-6.2467	1578.2	
7	.7	-5.7057	1314.1	
8	.8	-4.4055	908.05	
9	.9	-2.2164	457.67	
10	1	1	1	

Table 6.2:

Itération de Picard avec $\alpha = 1$, $\omega = 1$

x	y_0	y_1	y_2	y_3
0	0	0	0	0
7.698	.1	-.2240	1.7169	-13.747
8.391	.2	-.0340	1.4325	-10.283
8.796	.3	.1200	1.2537	- 7.796
9.084	.4	.2591	1.1379	- 5.867
9.307	.5	.3901	1.0625	- 4.289
9.489	.6	.5163	1.0156	- 2.953
9.643	.7	.6394	.9897	- 1.792
9.777	.8	.7606	.9804	- .764
9.895	.9	.8806	.9846	.156
1	1	1	1	1

Diverge!

Table 6.3:

Itération de Picard avec $\alpha = 1$, $\omega = .2$

x	y_0	y_1	y_2	y_3	y_{10}	y_{25}
0	0	0	0	0	0	0
7.698	.1	.0352	.0610	.0502	.0526	.052558
8.391	.2	.1532	.1745	.1675	.1726	.17305
8.796	.3	.2640	.2806	.2753	.2793	.27956
9.084	.4	.3718	.3845	.3802	.3826	.38271
9.307	.5	.4780	.4874	.4838	.4849	.48481
9.489	.6	.5833	.5899	.5870	.5870	.58673
9.643	.7	.6879	.6922	.6900	.6893	.68893
9.777	.8	.7921	.7946	.7930	.7921	.79173
9.895	.9	.8961	.8972	.8963	.8956	.89535
10	1	1	1	1	1	1

Pour $\alpha = 1$, nous avons répété l'itération de Picard partant avec les mêmes valeurs pour y_o. Le résultats figurent dans la table 6.2. On voit que les points du réseau ont été fortement décalés vers $x = 10$. Toujours la solution diverge après 4 pas. On remarque qu'après un pas la solution y_1 au point x_1 devient négative, ceci pour se faire corriger d'autant plus (mais dans l'autre sens!) un pas plus tard.

Ce phénomène de croissance oscillante peut être évité par une pondération de deux solutions consécutives y_{k+1} et y_k en calculant une nouvelle solution $\tilde{y}_{k+1}$ par

$$\tilde{y}_{k+1} = \omega\ y_{k+1} + (1-\omega)\ y_k \qquad (6.79)$$

Pour $\omega = 0.2$, les solutions y_k, après $k = 1,2,3,10$ et 25 pas sont représentées dans la table 6.3. On voit que l'on tend vers la bonne solution en oscillant autour d'elle. On obtient quatre décimales de la solution après 25 itérations. Le choix d'un $\omega = 0.3$ fait déjà diverger le processus.

La méthode d'itération de Picard est une méthode très simple pour la résolution de problèmes non linéaires. Son grand désavantage c'est que l'on ne peut pas assurer sa convergence pour tous les cas, car elle ne dépend pas seulement du type d'équation mais aussi du choix de la solution initiale et de la pondération.

6.5.3 METHODE DE NEWTON

La méthode de Newton est bien connue pour chercher le zéro d'une fonction $f(x) = 0$. On part d'une valeur approchée $x = x_0$ (voir Fig. 6.18). En x_0 on linéarise la fonction

$$f(x) \approx f(x_0) + (x-x_0)\ (df/dx)\ (x_0) = 0 \qquad (6.80)$$

donc

$$x = x_0 - f(x_0)\ /\ (df/dx)\ (x_0). \qquad (6.81)$$

La convergence de ce processus dépend du choix de la solution initiale x_0. Avec $\tilde{x}_0$ comme solution initiale (voir Fig. 6.18), la méthode de Newton risque de ne pas converger!

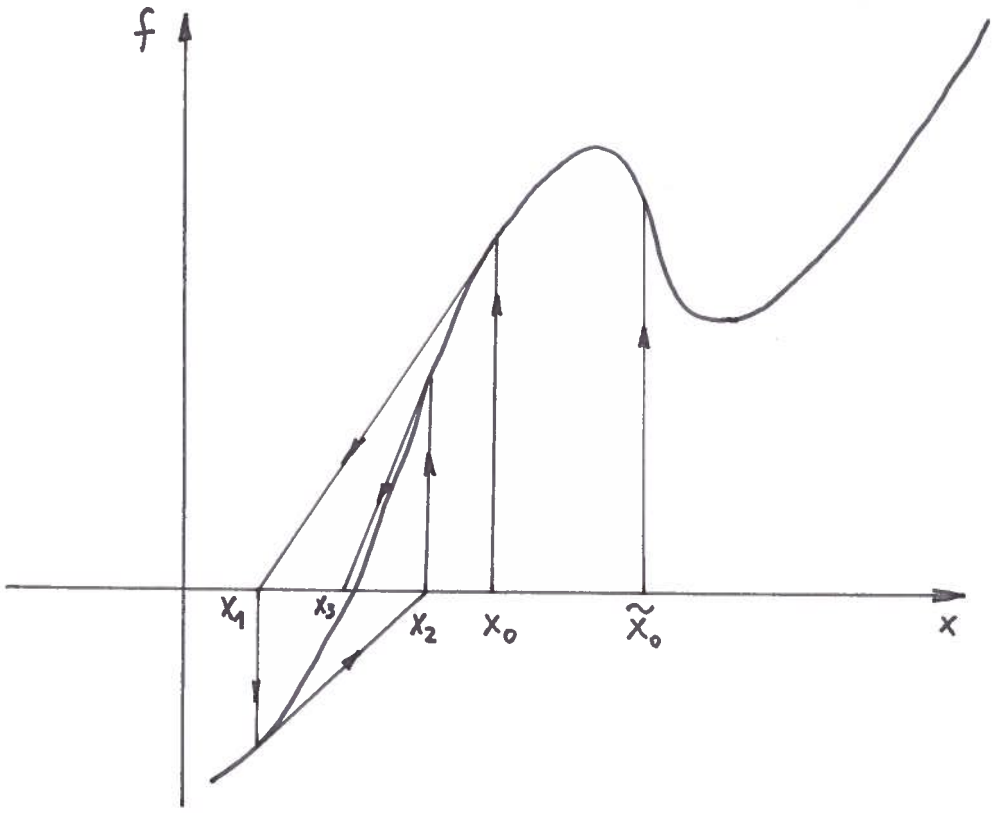


Fig. 6.18: Méthode de Newton pour trouver x tel que f(x) = 0. Avec $\tilde{x}_0$ comme solution de départ la méthode risque de diverger.

Appliquons maintenant cette méthode au problème (6.67) en partant de la même solution initiale $y_o = x/10$. Cette première approximation satisfait aux conditions aux limites (6.63). Dans le processus itératif ($k > 1$)

$$y_k(x) = y_{k-1}(x) + u_k(x) \qquad (6.82$$

où la fonction $u_k(x)$ joue le rôle d'une correction de l'approximation y_{k-1}. Puisque y_o satisfait aux conditions aux limites,

$$u_k(0) = u_k(10) = 0 \qquad (6.83)$$

Si le processus (6.82) tend vers la bonne solution (ce qui n'est pas garanti) on trouve

$$\lim_{k\to\infty} y_k = y \qquad (6.84)$$

et la correction

$$\lim_{k\to\infty} u_k = 0 \qquad (6.85)$$

On admet, pour un k donné, que la correction u_k soit petite, donc par exemple

$$|u_k| \ll \max_{0 \leq x \leq 10}(y_{k-1}(x)) - \min_{0 \leq x \leq 10}(y_{k-1}(x)) \qquad (6.86)$$

On linéarise l'équation pour le pas k

$$y''_k = y''_{k-1}+u''_k = \mathrm{Sh}\ y_k = \mathrm{Sh}(y_{k-1}+u_k) \approx \mathrm{Sh}\ y_{k-1}+u_k\ \mathrm{Ch}\ y_{k-1} \qquad (6.87)$$

L'équation différentielle linéarisée pour u_k s'écrit alors

$$u''_k - u_k\ \mathrm{Ch}\ y_{k-1} + (y''_{k-1} - \mathrm{Sh}\ y_{k-1}) = 0 \qquad (6.88)$$

où u_k satisfait aux conditions (6.83).

En multipliant l'équation différentielle linéarisée (6.88) par la fonction $\eta(x)$, puis en intégrant sur tout le domaine (intégration par partie du premier et du troisième terme) on obtient le problème suivant:

"Chercher $u_k \in V$, $u_k(o) = u_k(10) = 0$ tel que

$$\int_o^{10} (\eta' u_k' + \eta\, u_k \,\mathrm{Ch}\, y_{k-1} + \eta'\, y_{k-1}' + \eta\,\mathrm{Sh}\, y_{k-1})\, dx = 0 \qquad (6.89)$$

pour tout $\eta \in V$, $\eta(0) = \eta(10) = 0$"

La résolution du problème linéarisé (6.89) se fait par la méthode des éléments finis. Le domaine $0 \leq x \leq 10$ est subdivisé en N intervalles. La fonction u_k inconnue est représentée comme une combinaison linéaire de fonctions de forme $e_j(x)$ données

$$u_k(x) = \sum_{j=1}^{N-1} u_j\, e_j(x) \qquad (6.90)$$

Notons que les conditions aux limites $u_0 = 0$ et $u_N = 0$ sont satisfaites. Les éléments de base linéaires $e_j(x)$ sont définis par eq. (6.9) et représentés sur les figures 6.3 et 6.4. En introduisant les dépendances fonctionnelles des éléments de base et en choisissant $\eta_i \equiv e_i$ pour $i = 1,.., N-1$, la forme faible (6.89) s'écrit

$$\sum_{j=1}^{N-1} u_j \int_{x_{i-1}}^{x_{i+1}} (e_i' e_j' + e_i e_j \mathrm{Ch}\, y_{k-1})dx + \int (e_i' y_{k-1}' + e_i\, \mathrm{Sh}\, y_{k-1})dx = 0 \qquad (6.91)$$

$$i = 1,2,.. N - 1.$$

Le problème (6.91) devient un système de N-1 équations linéaires (i = 1,.., N-1) avec N-1 inconnues (les u_1 à u_{N-1}). Il s'écrit sous forme abrégée

$$\sum_{j=1}^{N-1} a_{ij}\, u_j + b_i = 0, \qquad i = 1,\ldots, N-1 \tag{6.92}$$

où

$$a_{ij} = \int_{x_{i-1}}^{x_{i+1}} (e_i'\, e_j' + e_i e_j \operatorname{Ch} y_{k-1})\, dx \tag{6.93}$$

et

$$b_i = \int_{x_{i-1}}^{x_{i+1}} (e_i'\, y_{k-1}' + e_i \operatorname{Sh} y_{k-1})\, dx. \tag{6.94}$$

L'équation (6.92) équivaut au problème

$$A\,\underline{u} + \underline{b} = 0 \tag{6.95}$$

En pratique, les éléments de la matrice A et du membre de droite $\underline{b}$ se calculent itérativement en ajoutant les contributions d'un intervalle après l'autre. Initialisons d'abord $A = \underline{b} = 0$ et le processus itératif s'écrit:

"De i = 1 à N effectue

$$a_{i-1\ i-1} = a_{i-1\ i-1} + \int_{x_{i-1}}^{x_i} \frac{dx}{(x_i - x_{i-1})^2} \left[1 + (x-x_i)^2 \operatorname{Ch} y_{k-1}\right]$$

$$a_{i-1\ i} = a_{i-1\ i} - \int_{x_{i-1}}^{x_i} \frac{dx}{(x_i - x_{i-1})^2} \left[1 + (x-x_{i-1})\,(x-x_i) \operatorname{Ch} y_{k-1}\right] \tag{6.96}$$

$$a_{i\ i} = a_{i\ i} + \int_{x_{i-1}}^{x_i} \frac{dx}{(x_i - x_{i-1})^2} \left[1 + (x-x_{i-1})^2 \operatorname{Ch} y_{k-1}\right]$$

et

$$b_{i-1} = b_{i-1} - \int_{x_{i-1}}^{x_i} \frac{dx}{x_i - x_{i-1}} \left[y'_{k-1} + (x - x_i)\ \mathrm{Sh}\ y_{k-1} \right]$$

$$b_i = b_i + \int_{x_{i-1}}^{x_i} \frac{dx}{x_i - x_{i-1}} \left[y'_{k-1} - (x - x_{i-1})\ \mathrm{Sh}\ y_{k-1} \right] \qquad (6.97)$$

Il faut noter que $a_{i-1\ i} = a_{i\ i-1}$. Les conditions aux limites $u_0 = u_N = 0$ s'imposent en biffant lignes et colonnes 0 et N de A et les éléments b_0 et b_N.

Cette procédure est utilisée dans le programme YPPSHY donné dans appendice 6A.

Ayant résolu le problème linéaire (6.95) il faut corriger la solution y d'après (6.82). Afin de pouvoir contrôler la convergence du processus on introduit un poids ω qui multiplie la correction u_k. L'équation de linéarisation (6.82) s'écrit alors

$$y_k(x) = y_{k-1}(x) + \omega\ u_k(x) \qquad (6.98)$$

avec $0 < \omega < 2$. En pratique, souvent $\omega < 1$. Si l'on est proche de la solution, la méthode de Newton ($\omega = 1$) double par pas le nombre de décimales convergés.

La méthode de résolution décrite à été essayée dans le programme YPPSHY. Nous avons choisi $\omega = 1$, un réseau équidistant en x et une solution initiale linéaire $y_0 = x/10$ qui satisfait aux conditions aux limites. Les résultats sont listés dans la Table 6.4. On constate que la solution est obtenue après 3 pas d'itération. Le résultat final est représenté à la figure 6.16. La convergence rapide est due au fait que la méthode des éléments finis est une méthode très rigide. Puisqu'on minimise pour chaque pas l'intégrale (6.91) on évite la formation d'instabilités numériques.

Table 6.4 :

Solutions itératives de y" = Sh y sans accumulation des points du réseau

x	y_0	y_1	y_2	y_3	y_4
0	0	0	0	0	0
1	.1	.00200	6.999-5	6.894-5	6.894-5
2	.2	.00583	2.240-4	2.206-4	2.206-4
3	.3	.01323	6.465-4	6.370-4	6.370-4
4	.4	.02581	.00184	.00182	.00182
5	.5	.04531	.00524	.00518	.00518
6	.6	.07432	.01490	.01476	.01476
7	.7	.11930	.04232	.04205	.04205
8	.8	.20130	.12022	.11984	.11984
9	.9	.39661	.34270	.34241	.34241
10	1	1	1	1	1

Table 6.5 :

Solution itérative de y" = Shy avec accumulation des points du réseau.

x	y_o	y_1	y_2	y_3
0	0	0	0	0
7.698	.1	.05262	.05256	.05256
8.391	.2	.17312	.17306	.17306
8.796	.3	.27960	.27956	.27956
9.084	.4	.38274	.38271	.38271
9.307	.5	.48483	.48480	.48480
9.489	.6	.58675	.58672	.58672
9.643	.7	.68894	.68893	.68893
9.777	.8	.79173	.79172	.79172
9.895	.9	.89534	.89534	.89534
10	1	1	1	1

Le même calcul a été effectué en accumulant les points du réseau en utilisant la fonction de densité (6.75) et $\alpha = 1$. Les résultats sont montrés dans la table 6.5. On voit que la solution initiale y_0 est déjà une très bonne approximation de la solution exacte. Comme conséquence, après un pas d'itération, la solution $y_1(x)$ est déjà précise à quatre décimales.

Nous avons essayé de faire dérailler la Méthode de Newton en choisissant comme solution initiale

$$y_0(x) = (x/10) * \cos(\pi x), \qquad (6.99)$$

mais sans succès. Il semble que, pour le poblème y" = Shy, la méthode de Newton soit stable pour toute solution initiale y_0. Cependant, cela n'est pas forcement vrai pour d'autres problèmes.

6.5.4 METHODE DE CONTINUATION

Si ni la méthode de Picard ni la méthode de Newton ne convergent, il faut faire appel à la méthode de continuation. Au lieu de résoudre y" = Shy, on résoud

$$f(y,\lambda) = y'' - \lambda \, Shy = 0 \qquad (6.100)$$

$$y(0) = 0, \quad y(10) = 1.$$

Le paramètre λ est introduit pour enclencher la nonlinéarité. On commence par exemple par une solution du problème linéaire:

$$y_0 = x/10, \quad \lambda_0 = 0. \qquad (6.101)$$

Ayant choisi un λ_1, on obtient la nouvelle solution y_1 en deux pas: D'abord on calcule une prédiction de y_1 (méthode d'Euler) par un développement de $f(\lambda,y)$ en une série de Taylor autour de (λ_o,y_o):

$$f(\lambda,y) = f(\lambda_o,y_o) - (\lambda-\lambda_o)\ Shy_o + (y-y_o)'' - \lambda_o Chy_o(y-y_o)$$

$$+ O\left((\lambda-\lambda_o)^2 + (y-y_o)^2\right) = 0 \qquad (6.102)$$

Puisque $f(\lambda_o,y_o) = 0$, le problème de prédiction pour la correction $\Delta y = y_1-y_o$, ayant choisi $\Delta\lambda = \lambda_1-\lambda_o$ devient

$$\Delta y'' - \Delta y\lambda_o Chy_o = \Delta\lambda\ Shy_o \qquad (6.103)$$

La solution de (6.103) livre une prédiction pour la solution $y_1 = y_o + \Delta y$ correspondant à une valeur de $\lambda = \lambda_1$.

Le pas suivant est la correction de y_1 par la méthode de Newton. La solution prédite est nommée y_1^o et la solution après un pas de Newton $y_1^1 = y_1^o + u_1$. L'itération de Newton s'écrit (voir eq. 6.88)

$$u''_{k+1} - u_{k+1}\ Chy_k + (y''_k - Shy_k) = 0. \qquad (6.104)$$

Nous avons ainsi trouvé un y_1 convergé correspondant à $\lambda = \lambda_2 > \lambda_1$, on effectue un pas de prédiction pour y_2 et une correction par la méthode de Newton. La solution du problème initial est obtenue lorsque $\lambda = 1$.

Ce processus est schématiquement décrit sur la Figure 6.19. On y représente la dépendance de la norme de y en fonction du paramètre λ. Les ronds (o) sont les valeurs prédites, les croix (x) correspondent à des valeurs convergées après une itération de Newton. Le point (•) sur la courbe en $\lambda = \lambda_R$ est un point de retournement. En $\lambda = \lambda_3$, la méthode de Newton risque de diverger. Comment arriver à $\lambda = 1$?

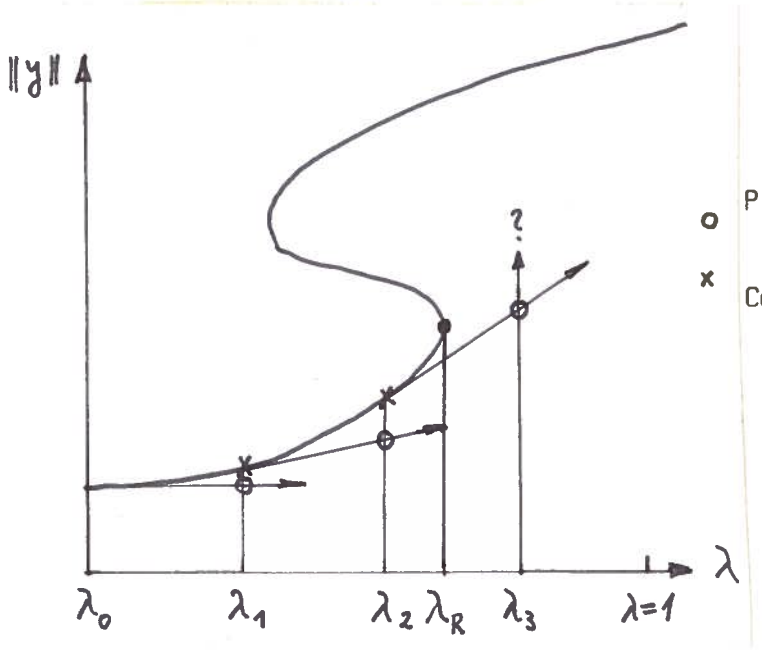


Fig. 6.19 : Dépendance schématique de la norme de y comme fonction de λ.

Au lieu de prendre λ comme paramètre on pourrait choisir la norme de y par exemple. On effectue un calcul du type itération inverse

$$y''_{k+1} = Shy_k \tag{6.105}$$

en normalisant après coup par

$$||y_{k+1}|| = R. \tag{6.106}$$

Cela implique que par le choix de R on a choisi un λ dans le problème

$$y'' = \lambda Shy \tag{6.107}$$

qui vaut

$$\lambda = \frac{R}{||y_{k+1}||} \tag{6.108}$$

Il faut donc ajuster R jusqu'à ce que λ = 1. Dans le cas présenté dans Fig. 6.19 l'utilisation de la norme de y comme paramètre dans la méthode de continuation élimine les problèmes des points de retournement.

6.6 PROBLEME NON LINEAIRE BIDIMENSIONNEL

6.6.1 CHOIX DU MEMBRE DE DROITE

Au chapitre 3 ainsi que dans les paragraphes 6.3 et 6.4, nous avons discuté la résolution numérique d'une équation différentielle à dérivées partielles avec un membre de droite donné. Ce cas, traitable analytiquement, est un bon exemple de test, mais ne représente pas la réalité physique. Afin de mieux représenter les mesures expérimentales, il faut choisir des fonctions non linéaires comme profils de $dp/d\psi$ et de $1/2\ dT^2/d\psi$ avec la condition que les deux fonctions s'annulent sur $\delta\Omega$. Nous avons porté notre choix sur des polynômes en ψ, de la forme

$$\frac{dp}{d\psi} = \sum_{l=1}^{L} p_l \left(\frac{\psi_s - \psi}{\psi_s}\right)^l$$

$$T\frac{dT}{d\psi} = \sum_{l=1}^{L} t_l \left(\frac{\psi_s - \psi}{\psi_s}\right)^l \qquad (6.109)$$

Il faut noter que la normalisation des profils par ψ_s permet de fixer la valeur au centre à

$$\frac{dp}{d\psi}(\psi = 0) = \sum_{l=1}^{L} p_l$$

$$T\frac{dT}{d\psi}(\psi = 0) = \sum_{l=1}^{L} t_l \qquad (6.110)$$

En introduisant la nouvelle variable $\tilde{\psi} = \psi_s - \psi$, nous sommes conduits à l'équation différentielle à dérivées partielle du problème de l'équilibre MHD (eq. 6.25)

$$L(\tilde{\psi}) = \frac{1}{r}\frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial\tilde{\psi}}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2\tilde{\psi}}{\partial z^2} = -S(r^2,\tilde{\psi})$$

$$= \sum_{l=1}^{L}\left(p_l + \frac{1}{r^2}t_l\right)\left(\frac{\tilde{\psi}}{\psi_s}\right)^l \qquad (6.111)$$

Cette équation est non linéaire si pour $l > 1$, $p_l \neq 0$ ou $t_l \neq 0$, ce qui est, en général, le cas.

Si dans l'expression du membre de droite $\tilde{\psi}$ n'était pas normalisé par ψ_s, toute méthode itérative pour résoudre (6.111) convergerait vers la solution triviale. En fixant les p_l et les t_l, on impose la solution vers laquelle un processus itératif converge. Souvent on ne prescrit pas entièrement les profils, mais on impose plutôt une grandeur physique telle que le courant total qui circule dans la décharge. Pendant l'itération, on demande donc que l'intégrale

$$I = \iint_{\Omega} S(r^2,\tilde{\psi})\, drdz \qquad (6.112)$$

reste fixe. Au lieu d'essayer de résoudre (6.111), on va s'attaquer au problème

$$L(\tilde{\psi}) = -\lambda\, S(r^2,\tilde{\psi}) \qquad (6.113)$$
$$\tilde{\psi}(\delta\Omega) = 0$$

avec

$$\lambda \iint_{\Omega} S(r^2,\tilde{\psi})\, drdz = I. \qquad (6.114)$$

6.6.2 LA METHODE DE PICARD

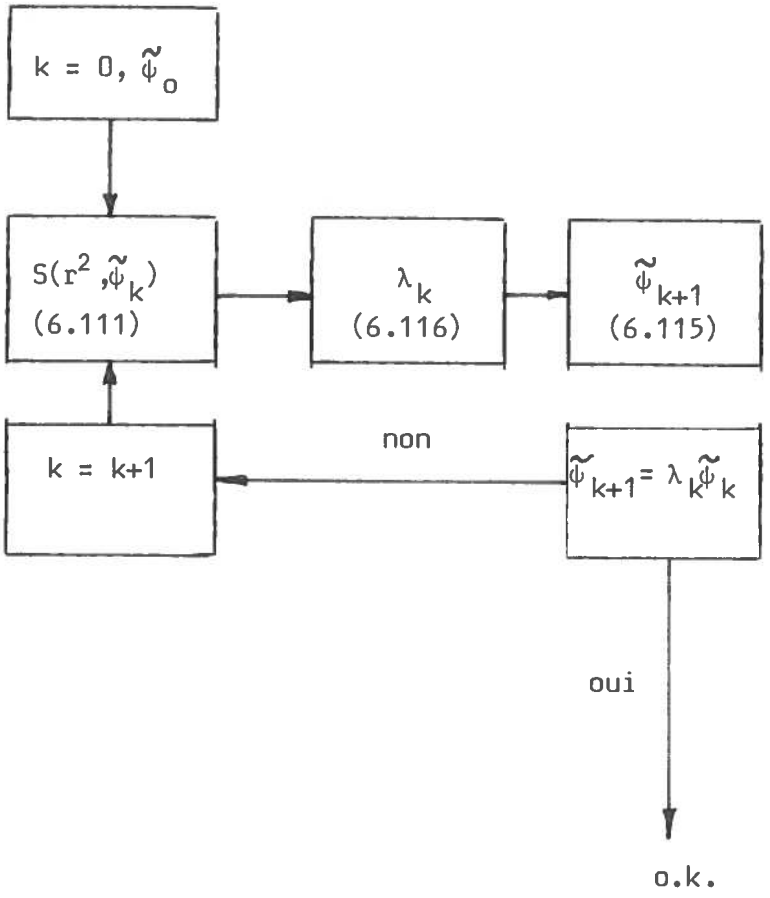
La méthode de Picard est une méthode itérative pour résoudre (6.113 et 6.114). Admettons qu'après k pas d'itérations, la solution soit $\tilde{\psi}_k$. Pour trouver $\tilde{\psi}_{k+1}$, on effectue

$$\begin{aligned} L(\tilde{\psi}_{k+1}) &= -\lambda_k S(r^2, \tilde{\psi}_k) \\ \tilde{\psi}_{k+1}(\delta\Omega) &= 0 \end{aligned} \qquad (6.115)$$

avec

$$\lambda_k \iint_\Omega S(r^2, \tilde{\psi}_k) = I. \qquad (6.116)$$

L'itération de Picard suit donc le schéma suivant:



6.6.3 METHODE DE NEWTON

La linéarisation ne doit se faire que sur le membre de droite:

$$S(r^2,\tilde{\psi}_{k+1}) \simeq S(r^2,\tilde{\psi}_k) + (\tilde{\psi}_{k+1} - \tilde{\psi}_k) \frac{\partial S}{\partial \tilde{\psi}} (\tilde{\psi}_k) \qquad (6.117)$$

où

$$\frac{\partial S}{\partial \tilde{\psi}} (\tilde{\psi}_k) = - \sum_{l=1}^{L} l \, \tilde{\psi}_k^{l-1} (p_l + \frac{1}{r^2} t_l). \qquad (6.118)$$

En introduisant la correction $\Delta\tilde{\psi} = \tilde{\psi}_{k+1} - \tilde{\psi}_k$, on obtient

$$L(\Delta\tilde{\psi}) + \lambda_k \Delta\tilde{\psi} \frac{\partial S}{\partial \tilde{\psi}} (\tilde{\psi}_k) = -L(\tilde{\psi}_k) - \lambda_k S(r^2,\tilde{\psi}_k). \qquad (6.119)$$

6.6.4 Méthode de continuation

En pratique, on part souvent d'une solution qui est un équilibre proche de celui que l'on veut calculer. La méthode de Newton converge donc presque toujours. Si l'estimation à l'aide d'un équilibre proche ne suffit pas, on peut calculer une estimation à l'aide d'une suite d'équilibres calculés auparavant. On connaît, par exemple, deux équilibres qui ne diffèrent que par un paramètre, disons par la valeur de p_3. Si l'on cherche un troisième équilibre avec une troisième valeur de p_3, et tous les autres paramètres fixés, on peut extrapoler une solution de départ à l'aide de ces deux équilibres déjà connus.

Une autre possibilité serait de partir d'une solution triviale $\lambda_0 = 0$, $\psi_0 = 0$ et d'enclencher lentement le courant I ou λ. On suit la branche de solution jusqu'à ce qu'on atteigne la valeur exigée pour I.

6.7 DISCUSSION

Dans ce chapitre 6 sur les équations différentielles à dérivées partielles, nous avons seulement discuté les équations elliptiques. Ce sont les équations dont une perturbation locale engendre une modification globale de la solution. L'exemple particulier de l'équilibre MHD a été choisi. La méthode des différences finies a été appliquée au § 3 pour préparer un problème matriciel qui a été résolu par différentes méthodes.

Dans ce chapitre 6, nous avons appliqué la méthode des éléments finis pour résoudre numériquement ce même problème. Malgré les avantages que les éléments finis ont vis-à-vis des différences finies, cette méthode n'est guère utilisée par les physiciens. Et pourtant ses avantages sont multiples:

- par des intégrations par partie on réduit l'ordre des dérivées les plus élevées;
- par le fait que l'on a un problème d'optimisation, on évite les solutions oscillantes si on traite des problèmes non linéaires, donc on n'a pas de difficulté due à la stabilité numérique;
- les conditions aux limites sont introduites tout à fait naturellement puisqu'on peut placer des points du réseau exactement sur le bord;
- on n'est pas obligé de choisir un réseau équidistant pour conserver le même ordre de convergence. Par le fait que l'on a pu réduire l'ordre des dérivées on peut choisir un réseau adapté au problème non équidistant sans perdre l'ordre de convergence;
- les conditions aux limites du type Neumann (on prescrit la dérivée) deviennennt des conditions dites naturelles. Par l'intégration par partie, on arrive à les imposer en laissant tomber la partie intégrée;
- le problème matriciel résultant d'une approximation numérique d'un opérateur symétrique et défini positif par la méthode des éléments finis reste symétrique et défini positif, ou autrement dit, un problème variationnel devrait être résolu en tant que problème variationnel;
- on obtient la solution partout dans le domaine, donc pas d'interpolations nécessaires.

La méthode des éléments finis semble avoir un inconvénient important: Elle semble être compliquée à programmer. En donnant les deux exemples de programmes dans les appendices 6A et 6B, nous essayons de familiariser le physicien avec l'organisation d'un code basé sur les éléments finis.

Nous n'avons pas pu discuter les problèmes du type évolution temporelle. Les problèmes lors d'une résolution numérique de ces équations paraboliques (problèmes de diffusion comme la conduction thermique) ou hyperboliques (problème de propagation d'ondes) ou les termes d'advection (dans des problèmes de transport dans une représentation eulérienne) ont été merveilleusement présentés par K.V. Roberts lors de son cours AVCP en 1973. Nous vous conseillons d'étudier ce papier.

6.8 BIBLIOGRAPHIE

Le livre classique sur la méthode des différences finies pour des problèmes à évolution temporelle:
R.D. Richtmeyer, K.W. Morton "Difference Methods for Initial Value Problems" (1967, John Wiley + Sons, New York).

Une réduction de ces 400 pages sur 50 dactylographiées contenant tout ce qui intéresse le physicien a été donne dans:
K.V. Roberts, "Numerical Solution of Initial-Value Problems in Classical Physics", XV^{e} Cours de perfectionnement 1973.

Depuis lors, de grands progrès ont été faits, spécialement dans le traitement des termes advectifs. La littérature à proposer:

J.P. Boris, D.L. Book, "Flux Corrected Transport", J. Comp. Phys. 20 (1976) 397.

Sur le problème variationnel traité par éléments finis il y a le livre:
G. Strang, G.J. Fix, "An Analysis of the Finite Element Method", (1973, Prentice Hall).

Il contient surtout le traitement des problèmes elliptiques et un petit chapitre sur les problèmes à valeurs initiales où le temps est discrétisé par des différences finies et l'opérateur spatial par des éléments finis.

Les méthodes de continuation et de bifurcation sont discutés dans:

H.B. Keller "Numerical Solution of Bifurcation and Nonlinear Eigenvalue Problems" in "Applications of Bifurcation Theory (1977, P. Rabinowitz ed., Academic Press, N.Y.), pages 359-384.

Appendice 6A Programme YPPSHY

On y résoud l'équation différentielle ordinaire y" = Sh y soumis aux conditions aux limites y(0) = 0 et y(10) = 1 par la méthode des éléments finis et de Newton. Dans la sousroutine MESH une accumulation de points du réseau est faite vers le bord x = 10. Dans INIT on choisit la solution initiale y_0, dans CONMAT on construit la matrice A, dans RHS le membre de droite $\underline{b}$, dans BOUND on impose les conditions aux limites, dans AXB on résoud le système $A\underline{x} = \underline{b}$ et dans ALC on ajoute la correction Δy.

```
*COMDECK COMESH
      COMMON /COMESH/
     I  N,        N1,       N2,
     R  X(101),   XO,       DX(100),  XN

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN COMPHY

*COMDECK COMPHY
      COMMON /COMPHY/
     R  Y(201),   YO,       YN,       SH(201),  CH(201),  YP(100),
     R  OMEGA,    EPS,
     I  NSING

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN COMBLA

*COMDECK COMBLA
      COMMON
     R  A(202),   B(101)

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN MAIN

*DECK MAIN
      PROGRAM YPPSHY(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
C
C     SOLVES  D2Y/DX2 = SH(Y)  BY FINITE ELEMENTS
C
C     FORM THE MESH
C
      CALL MESH
      CALL OUTPUT(1)
C
C      INITIAL GUESS
C
      CALL INIT
      CALL OUTPUT(2)
C
C     ITERATION
C
      ITER=0
  100 CONTINUE
      ITER=ITER+1
      IF (ITER .GT. 10) STOP
C
C     CONSTRUCT THE MATRIX A
C
      CALL CONMAT
C
C     RIGHT HAND SIDE VECTOR B
C
      CALL RHS
C
C     BOUNDARY CONDITIONS
C
      CALL BOUND
C
C     SOLVE SYSTEM OF LINEAR EQUATIONS
C
```

```
            CALL AXB
C
C       ADD LINEAR CORRECTION
C
            CALL ALC
            CALL OUTPUT(3)
C
C       CONTROLE CONVERGENCE
C
            CALL CONVER(ICONV)
            IF (ICONV .EQ. 0) GO TO 100
C
C       FINAL SOLUTION
C
            CALL OUTPUT(4)
C
            STOP
            END

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN SUB1

*DECK SUB1
            SUBROUTINE MESH
*CALL COMESH
C
C       SET MESH QUANTITIES
C
            N=10
            N1=N+1
            N2=2*N+1
            X0=0.0
            XN=10.0
            ALFA=1.0
            AN=(EXP(ALFA*10.0)-1.0)/N
C
C        X AT MESH- AND INTERMEDIATE POINTS
C
            X(1)=X0
            DO  100  J=1,N
            X(2*J+1)=ALOG(AN+EXP(ALFA*X(2*J-1)))/ALFA
            X(2*J)=(X(2*J+1)+X(2*J-1))/2.0
            DX(J)=X(2*J+1)-X(2*J-1)
 100        CONTINUE
C
            RETURN
            END

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN SUB2

*DECK SUB2
            SUBROUTINE INIT
*CALL COMESH
*CALL COMPHY
C
C       INITIALIZES CONSTANTS
C
            OMEGA=1.0
```

```
      YO=0.0
      YN=1.0
      EPS=1.E-6
      ZDY=(YN-YO)/N
C
C     INITIAL GUESS
C
      Y(1)=YO
      DO  100  J=1,N
      Y(2*J+1)=Y(2*J-1)+ZDY
      Y(2*J)=(Y(2*J+1)+Y(2*J-1))/2.0
      YP(J)=(Y(2*J+1)-Y(2*J-1))/DX(J)
 100  CONTINUE
C
      RETURN
      END
```

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN SUB3

```
*DECK SUB3
      SUBROUTINE CONMAT
*CALL COMESH
*CALL COMPHY
*CALL COMBLA
C
C     CLEAR MATRIX A
C
      CALL VZERO(A,2*N1)
C
C     CALCULATE HYPERBOLIC FUNCTIONS
C
      DO  100  J=1,N2
      ZEXP=EXP(Y(J))
      ZEXM=1.0/ZEXP
      SH(J)=(ZEXP-ZEXM)/2.0
      CH(J)=(ZEXP+ZEXM)/2.0
 100  CONTINUE
C
C     SCAN OVER ALL N INTERVALS
C
      DO  200 J=1,N
C
C     SIMPSON:  LEFT POINT INTEGRANDS
C
      ZLI1 =(1.0+CH(2*J-1)*DX(J)**2)/DX(J)**2
      ZLI12=1.0/DX(J)**2
      ZLI2 =ZLI12
C
C     SIMPSON:  MID POINT INTEGRANDS
C
      ZMI1 =(1.0+CH(2*J)*DX(J)**2/4.0)/DX(J)**2
      ZMI12=(1.0-CH(2*J)*DX(J)**2/4.0)/DX(J)**2
      ZMI2 =ZMI1
C
C     SIMPSON:  RIGHT POINT INTEGRANDS
C
      ZRI1 =1.0/DX(J)**2
```

```
            ZRI12=ZRI1
            ZRI2 =(1.0+CH(2*J+1)*DX(J)**2)/DX(J)**2
C
C       INTEGRATE
C
            A(2*J-1)=A(2*J-1)+DX(J)*(ZLI1+4.0*ZMI1+ZRI1)/6.0
            A(2*J)  =A(2*J)-DX(J)*(ZLI12+4.0*ZMI12+ZRI12)/6.0
            A(2*J+1)=A(2*J+1)+DX(J)*(ZLI2+4.0*ZMI2+ZRI2)/6.0
C
  200       CONTINUE
C
            RETURN
            END

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN SUB4

*DECK SUB4
            SUBROUTINE RHS
*CALL COMESH
*CALL COMPHY
*CALL COMBLA
C
C       CLEAR RHS VECTOR
C
            CALL VZERO(B,N1)
C
C       SCAN OVER ALL N INTERVALS
C
            DO  200  J=1,N
C
C       SIMPSON:  LEFT POINT INTEGRANDS
C
            ZLI1=(YP(J)-SH(2*J-1)*DX(J))/DX(J)
            ZLI2=YP(J)/DX(J)
C
C       SIMPSON:  MID POINT INTEGRANDS
C
            ZMI1=(YP(J)-SH(2*J)*DX(J)/2.0)/DX(J)
            ZMI2=(YP(J)+SH(2*J)*DX(J)/2.0)/DX(J)
C
C       SIMPSON:  RIGHT POINT INTEGRANDS
C
            ZRI1=YP(J)/DX(J)
            ZRI2=(YP(J)+SH(2*J+1)*DX(J))/DX(J)
C
C       INTEGRATE
C
            B(J)=B(J)+DX(J)*(ZLI1+4.0*ZMI1+ZRI1)/6.0
            B(J+1)=B(J+1)-DX(J)*(ZLI2+4.0*ZMI2+ZRI2)/6.0
C
  200       CONTINUE
C
            RETURN
            END

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN SUB5
```

```
*DECK SUB5
      SUBROUTINE BOUND
*CALL COMESH
*CALL COMPHY
*CALL COMBLA
C
C     PUT FIRST VARIBLE TO ZERO
C
      A(1)=1.0
      A(2)=0.0
      B(1)=0.0
C
C     PUT LAST VARIABLE TO ZERO
C
      A(2*N)  =0.0
      A(2*N+1)=1.0
      B(N+1)=0.0
C
      RETURN
      END

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN SUB6

*DECK SUB6
      SUBROUTINE AXB
*CALL COMESH
*CALL COMPHY
*CALL COMBLA
C
C      SOLVES  A * X = B
C
C     DECOMPOSE A
C
      NSING=0
      CALL ALDLT(A,1,E-14,N,2,NSING)
      IF (NSING .NE. -1) GO TO 100
      CALL OUTPUT(5)
      STOP
 100  CONTINUE
C
C     SOLVE  L * D * LT * X = B
C
      CALL LYV(A,B,N,2,1)
      CALL DWY(A,B,N,2,1)
      CALL LTXW(A,B,N,2,1)
C
      RETURN
      END

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN SUB7

*DECK SUB7
      SUBROUTINE ALC
*CALL COMESH
*CALL COMPHY
*CALL COMBLA
C
```

```
C       ADD THE CORRECTION
C
            CALL SAXPY(N1,OMEGA,B,1,Y(1),2)
C
C       PRODUCE MID POINTS
C
            DO  100  J=1,N
            Y(2*J)=(Y(2*J-1)+Y(2*J+1))/2.0
 100        CONTINUE
C
C       NEW DERIVATIVES YP
C
            DO  200  J=1,N
            YP(J)=(Y(2*J+1)-Y(2*J-1))/DX(J)
 200        CONTINUE
C
            RETURN
            END

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN SUB8

*DECK SUB8
            SUBROUTINE CONVER(KCONV)
*CALL COMESH
*CALL COMPHY
*CALL COMBLA
C
C       NORM OF SOLUTION Y
C
            ZNORM=SQRT(SDOT(N1,Y(1),2,Y(1),2))
C
C       NUMBER OF CONVERGED COMPONENTS
C
            IC=0
            DO  100  J=1,N1
            IF (ABS(B(J))/ZNORM .LT. EPS) IC=IC+1
 100        CONTINUE
C
C       ALL COMPONENTS CONVERGED OR NOT
C
            KCONV=0
            IF (IC .EQ. N1) KCONV=1
C
            RETURN
            END

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN SUB9

*DECK SUB9
            SUBROUTINE OUTPUT(K)
*CALL COMESH
*CALL COMPHY
*CALL COMBLA
C
            GO TO (100,200,300,400,500),K
C
C       MESH
```

```
C
 100      CONTINUE
          CALL PAGE
          CALL IVAR(8HN    =      ,N)
          CALL RARRAY(8HXMESH = ,X,N2)
C
          RETURN
C
C      INITIAL CONDITIONS
C
 200      CONTINUE
          CALL RVAR(8HYO =      ,YO)
          CALL RVAR(8HYN =      ,YN)
          CALL RVAR(8HOMEGA = ,OMEGA)
          CALL RVAR(8HEPS =     ,EPS)
          CALL RARRAY(8HYINIT = ,Y,N2)
C
          RETURN
C
C      PERIODIC OUTPUT
C
 300      CONTINUE
          CALL RARRAY(8HUK =      ,B,N1)
          CALL RARRAY(8HY  =      ,Y,N2)
C
          RETURN
C
C      FINAL SOLUTION
C
 400      CONTINUE
          CALL RARRAY(8HXMESH = ,X,N2)
          CALL RARRAY(8HY  =      ,Y,N2)
          RETURN
C
C      MATRIX A IS SINGULAR
C
 500      CONTINUE
          CALL IVAR(8HNSING = ,NSING)
          CALL RARRAY(8HA  =      ,A,2*N1)
C
          RETURN
          END

0         1         2         3         4         5         6         7
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789
```

```
N =     =          10

XMESH =
   0.          3.8489E+00  7.6978E+00  8.0443E+00  8.3907E+00  8.5934E+00  8.7961E+00  8.9400E+00  9.0838E+00  9.1953E+00
 9.3069E+00    9.3981E+00  9.4892E+00  9.5663E+00  9.6433E+00  9.7101E+00  9.7769E+00  9.8358E+00  9.8946E+00  9.9473E+00
 1.0000E+01

YO =      =     0.

YN =      =     1.0000E+00

OMEGA =   =     1.0000E+00

EPS =     =     1.0000E-06

YINIT =
   0.          5.0000E-02  1.0000E-01  1.5000E-01  2.0000E-01  2.5000E-01  3.0000E-01  3.5000E-01  4.0000E-01  4.5000E-01
 5.0000E-01    5.5000E-01  6.0000E-01  6.5000E-01  7.0000E-01  7.5000E-01  8.0000E-01  8.5000E-01  9.0000E-01  9.5000E-01
 1.0000E+00

UK =
   0.          -4.7384E-02 -2.6884E-02 -2.0396E-02 -1.7257E-02 -1.5168E-02 -1.3255E-02 -1.1058E-02 -8.2714E-03 -4.6562E-03
   0.

Y  =
   0.          2.6306E-02  5.2616E-02  1.1287E-01  1.7312E-01  2.2636E-01  2.7960E-01  3.3117E-01  3.8274E-01  4.3379E-01
 4.8483E-01    5.3579E-01  5.8675E-01  6.3784E-01  6.8894E-01  7.4034E-01  7.9173E-01  8.4354E-01  8.9534E-01  9.4767E-01
 1.0000E+00

UK =
   0.          -5.8501E-05 -5.3622E-05 -4.5052E-05 -3.7268E-05 -3.0056E-05 -2.3228E-05 -1.6742E-05 -1.0655E-05 -5.0575E-06
   0.

Y  =
   0.          2.6279E-02  5.2557E-02  1.1281E-01  1.7306E-01  2.2631E-01  2.7956E-01  3.3113E-01  3.8271E-01  4.3375E-01
 4.8480E-01    5.3576E-01  5.8672E-01  6.3782E-01  6.8893E-01  7.4032E-01  7.9172E-01  8.4353E-01  8.9534E-01  9.4767E-01
 1.0000E+00

UK =
   0.          -8.2788E-11 -1.2526E-10 -1.2598E-10 -1.1107E-10 -9.0795E-11 -6.9542E-11 -4.9322E-11 -3.0945E-11 -1.4573E-11
   0.

Y  =
   0.          2.6279E-02  5.2557E-02  1.1281E-01  1.7306E-01  2.2631E-01  2.7956E-01  3.3113E-01  3.8271E-01  4.3375E-01
 4.8480E-01    5.3576E-01  5.8672E-01  6.3782E-01  6.8893E-01  7.4032E-01  7.9172E-01  8.4353E-01  8.9534E-01  9.4767E-01
 1.0000E+00

XMESH =
   0.          3.8489E+00  7.6978E+00  8.0443E+00  8.3907E+00  8.5934E+00  8.7961E+00  8.9400E+00  9.0838E+00  9.1953E+00
 9.3069E+00    9.3981E+00  9.4892E+00  9.5663E+00  9.6433E+00  9.7101E+00  9.7769E+00  9.8358E+00  9.8946E+00  9.9473E+00
 1.0000E+01

Y  =
   0.          2.6279E-02  5.2557E-02  1.1281E-01  1.7306E-01  2.2631E-01  2.7956E-01  3.3113E-01  3.8271E-01  4.3375E-01
 4.8480E-01    5.3576E-01  5.8672E-01  6.3782E-01  6.8893E-01  7.4032E-01  7.9172E-01  8.4353E-01  8.9534E-01  9.4767E-01
 1.0000E+00
```

Appendice 6B Sousroutines pour la construction de A

Nous essayons de montrer l'avantage d'une programmation modulaire. Dans AMATRX. on construit la matrice A. D'abord on choisit les éléments de base dans BASIS. Pour les deux termes quadratiques du problème de l'équilibre MHD, on définit la physique dans CONST, crée le vecteur $\underline{v}$ dans VECT et calcule la contribution à la matrice A provenant d'un point de Gauss par une multiplication diadique dans DIADIC.

```
*DECK C2SB10
      SUBROUTINE INTEGA(KR,KZ)
C
C
C  C2SB10  COMPUTE CELL CONTRIBUTION TO MATRIX A
C
C
*CALL COMINT
*CALL COMNJM
C
C     *    *    *    *    *    *    *    *    *    *    *    *    *    *
C
      CALL RESETR(XAB,NVAL*NVAL,0.0)
C
C     DEFINE THE CELL
C
      CALL DEFCEL(KR,KZ)
C
C     DEFINE INTEGRATION POINTS AND ASSOCIATED WEIGTHS
C
      CALL DEFINT
C
C     LOOP OVER INTEGRATION POINTS
C
      DO 100 JP=1,NPOINT
C
C     COMPUTE CELL CONTRIBUTION
C
      CALL AMATRX(KR,KZ,JP)
C
C     PERFORM INTEGRATION
C
      DO 20 JROW=1,NVAL
      DO 10 JCOL=JROW,NVAL
      XAB(JCOL,JROW)=XAB(JCOL,JROW)+CDR*CDZ*CW(JP)*XT(JCOL,JROW)
   10 CONTINUE
   20 CONTINUE
C
  100 CONTINUE
C
      RETURN
      END
LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN C2SB11
*DECK C2SB11
      SUBROUTINE AMATRX(KR,KZ,KPOINT)
C
C
C  C2SB11  COMPUTE ELEMENTARY CELL CONTRIBUTION TO MATRIX A
C
C
*CALL COMINT
C
C     *    *    *    *    *    *    *    *    *    *    *    *    *    *
C
      DIMENSION ZC(4),    ZF(12),   ZV(4)
```

```
C
C       SET CONTRIBUTION MATRIX TO ZERO
C
            CALL RESETR(XT,NVAL*NVAL,0.0)
C
C       COMPUTE BASIS FUNCTIONS
C
            CALL BASIS(RRINT(KPOINT),RZINT(KPOINT),RRCEL,RZCEL,ZF)
C
C       SCAN OVER QUADRATIC TERMS
C
            DO 100 JT=1,2
C
C       COMPUTE CONSTANTS
C
            CALL CONST(JT,RRINT(KPOINT),ZC)
C
C       EXPRESS QUADRATIC TERM AS A VECTOR
C
            CALL VECT(JT,ZC,ZF,ZV)
C
C       DIADIC MULTIPLICATION
C
            CALL DIADIC(NVAL,ZC,ZV,XT)
C
  100       CONTINUE
C
            RETURN
            END

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN C2SB30

*DECK C2SB30
            SUBROUTINE BASIS(PR,PZ,PAR,PAZ,PF)
C
C
C C2SB30  COMPUTE FIRST ORDER BASIS FUNCTIONS
C
C
C     *     *     *     *     *     *     *     *     *     *     *     *     *     *
C
            DIMENSION PAR(2),   PAZ(2),   PF(12)
C
C       DEFINE SURFACE OF ELEMENTARY CELL
C
            ZSURF=1.0/((PAR(2)-PAR(1))*(PAZ(2)-PAZ(1)))
C
C       TERM :  PSI
C
            PF( 1)= ZSURF*(PR-PAR(2))*(PZ-PAZ(2))
            PF( 2)=-ZSURF*(PR-PAR(1))*(PZ-PAZ(2))
            PF( 3)=-ZSURF*(PR-PAR(2))*(PZ-PAZ(1))
            PF( 4)= ZSURF*(PR-PAR(1))*(PZ-PAZ(1))
C
C       TERM :  DPSI/DR
C
            PF( 5)= ZSURF*(PZ-PAZ(2))
```

```
            PF( 6)=-ZSURF*(PZ-PAZ(2))
            PF( 7)=-ZSURF*(PZ-PAZ(1))
            PF( 8)= ZSURF*(PZ-PAZ(1))
C
C       TERM :  DPSI/DZ
C
            PF( 9)= ZSURF*(PR-PAR(2))
            PF(10)=-ZSURF*(PR-PAR(1))
            PF(11)=-ZSURF*(PR-PAR(2))
            PF(12)= ZSURF*(PR-PAR(1))
C
            RETURN
            END

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN C2SB40

*DEC< C2SB40
            SUBROUTINE CONST(KTERM,PR,PC)
C
C
C  C2SB40  CONSTANT FOR QUADRATIC TERMS
C
C
C     *     *     *     *     *     *     *     *     *     *     *     *     *     *
C
            DIMENSION PC(4)
C
            GO TO (100,200,300) KTERM
C
C       CONSTANT FOR FIRST SQUARE
C
  100       CONTINUE
            PC(1)=PR
            PC(2)=1.0/PR
            RETURN
C
C       CONSTANT FOR SECOND SQUARE
C
  200       CONTINUE
            PC(1)=PR
            PC(3)=1.0/PR
            RETURN
C
C       CONSTANT FOR RIGHT HAND SIDE
C
  300       CONTINUE
            PC(4)=-SOURCE(PR)/PR
            RETURN
C
            END

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN C2SB41

*DEC< C2SB41
            SUBROUTINE VECT(KTERM,PC,PF,PV)
C
C
```

```
C C2SB41  QUADRATIC TERM AS VECTOR**2
C
C
C     *     *     *     *     *     *     *     *     *     *     *     *     *     *
C
            DIMENSION PC(4),     PF(12),     PV(4)
C
            GO TO (100,200,300) KTERM
C
C       TERM :  DPSI/DR
C
  100       CONTINUE
            PV(1)=PC(2)*PF( 5)
            PV(2)=PC(2)*PF( 6)
            PV(3)=PC(2)*PF( 7)
            PV(4)=PC(2)*PF( 8)
            RETURN
C
C       TERM :  DPSI/DZ
C
  200       CONTINUE
            PV(1)=PC(3)*PF( 9)
            PV(2)=PC(3)*PF(10)
            PV(3)=PC(3)*PF(11)
            PV(4)=PC(3)*PF(12)
            RETURN
C
C       TERM :  PSI
C
  300       CONTINUE
            PV(1)=PC(4)*PF( 1)
            PV(2)=PC(4)*PF( 2)
            PV(3)=PC(4)*PF( 3)
            PV(4)=PC(4)*PF( 4)
            RETURN
C
            END

LIST OF CONTROL, ACTIVE, AND/OR INACTIVE CARDS IN C2SB42

*DECK C2SB42
            SUBROUTINE DIADIC(KVAL,PC,PV,PX)
C
C
C C2SB42  DIADIC MULTIPLICATION :    PX = PX + PC(1)*PV*PV
C
C
C     *     *     *     *     *     *     *     *     *     *     *     *     *     *
C
            DIMENSION PC(KVAL),     PV(KVAL),     PX(KVAL,KVAL)
C
            DO 110 JROW=1,KVAL
            DO 100 JCOL=JROW,KVAL
            PX(JCOL,JROW)=PX(JCOL,JROW)+PC(1)*PV(JROW)*PV(JCOL)
  100       CONTINUE
  110       CONTINUE
C
```

```
      RETURN
      END
```

```
0         1         2         3         4         5         6         7
12345678901234567890123456789012345678901234567890123456789012345678901234567890
0.UCLP, IN, CRPPHP ,      0.732KLNS.
```